Gonçalo José Soares Dias Mestre

Licenciatura em Engenharia Electrotécnica e Computadores

Cliente *mobile* para consumo e disponibilização de serviços georreferenciados

Dissertação para obtenção do Grau de Mestre em
Engenharia Electrotécnica e Computadores

Orientador: Tiago Cardoso, Professor Auxiliar, DEE – FCT/UNL

Júri:

Presidente: Prof. Doutor João Miguel Murta Pina

Arguente: Prof. Doutor João Manuel Pereira Barroso

Vogais: Prof. Doutor Tiago Oliveira Machado de Figueiredo Cardoso

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Março de 2015

Gonçalo José Soares Dias Mestre

Licenciatura em Engenharia Electrotécnica e Computadores

Cliente *mobile* para consumo e disponibilização de serviços georreferenciados.

Dissertação para obtenção do Grau de Mestre em
Engenharia Electrotécnica e Computadores

Orientador: Prof. Doutor Tiago Oliveira Machado de Figueiredo Cardoso, Professor Auxiliar da Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Constituição do júri

Presidente: Prof. Doutor João Miguel Murta Pina, Professor Auxiliar da Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Arguente: Prof. Doutor João Manuel Pereira Barroso, Professor Associado com Agregação da Universidade de Trás-os-Montes e Alto Douro

Vogal: Prof. Doutor Tiago Oliveira Machado de Figueiredo Cardoso, Professor Auxiliar da Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Março de 2015

Cliente *mobile* para consumo e disponibilização de serviços georreferenciados.

Copyright © 2015. Todos os direitos reservados. Gonçalo José Soares Dias Mestre, Faculdade de Ciências e Tecnologia e Universidade Nova de Lisboa.

A Faculdade de Ciências e Tecnologia e a Universidade Nova de Lisboa têm o direito, perpétuo e sem limites geográficos, de arquivar e publicar esta dissertação através de exemplares impressos reproduzidos em papel ou de forma digital, ou por qualquer outro meio conhecido ou que venha a ser inventado, e de a divulgar através de repositórios científicos e de admitir a sua cópia e distribuição com objectivos educacionais ou de investigação, não comerciais, desde que seja dado crédito ao autor e editor.

Agradecimentos

Gostaria de agradecer ao Professor Tiago Cardoso pela oportunidade, pelo apoio e principalmente pela motivação ao longo da dissertação e ao Engenheiro Rui Pedro Henriques.

À Universidade Nova de Lisboa e à Faculdade de Ciências e Tecnologias por me terem acolhido e disponibilizado as condições que permitiram a minha formação. Esta casa permitiu-me crescer a todos os níveis, deu-me a conhecer muitas pessoas e a travar grandes amizades. Um muito obrigado a todos os docentes que contribuíram para a minha formação. A todos os meus colegas de faculdade que me acompanharam, em especial aos vários *testers* que ao longo desta dissertação foram criticando, incentivando e colaborando para o produto final.

À empresa *IrRADIARE* por me acolher de braços abertos durante um ano, em especial ao Eng. Rui Pedro Henriques, cujos ensinamentos, conhecimento, disponibilidade, amizade e experiência ajudaram-me bastante para alcançar os objectivos propostos.

Aos meus pais, Norberto e Laura, pelo apoio e pela possibilidade do percurso académico, ao meu irmão Henrique pela colaboração nas áreas audiovisuais e a todos os meus restantes familiares.

Por fim, um grande agradecimento aos meus amigos que me acompanham desde sempre pelos grandes momentos que passamos juntos e por estarem sempre lá quando são precisos.

Resumo

Actualmente as aplicações móveis são indispensáveis para o utilizador. De acordo com estudos recentes, as receitas geradas pela sua utilização resultaram em aproximadamente 20 biliões de dólares só em 2013. Com estes valores estima-se que no futuro estes números aumentem, daí tornar-se imperativo abordar num estudo científico sobre este mercado em expansão.

A somar às inúmeras aplicações referentes ao sistema móvel *Android*, surgem as aplicações de serviços georreferenciados que vão aqui ser abordadas em maior profundidade, nomeadamente o consumo e disponibilização destes serviços em clientes *mobile*.

Ponderadas diversas soluções é possível com alguns melhoramentos atingir benefícios específicos para um maior número de utilizadores tendo em conta as suas características *a priori*. Uma das questões principais passa pela capacidade de publicação de diversas fontes de informação (vários *publishers*) com o uso de uma *Application Programing Interface* (*API*) documentada e aberta para que qualquer entidade possa publicar os seus conteúdos georreferenciados. O consumo por parte dos utilizadores (*subscriber*) é feita numa plataforma móvel *Android* onde os possam vários *publishers* possam publicar os seus conteúdos a partir da posição do utilizador.

Esta publicação será facilitada criando bases de dados para o armazenamento de todas as informações relevantes tanto da parte dos *publishers* como dos *subscribers*.

A validação deste trabalho consistiu em testar vários tipos de publicações, nomeadamente texto, imagem e gráficos na sua pesquisa e consumo numa plataforma móvel *Android*. Numa segunda fase, foi testada a publicação de gráficos em tempo real na plataforma.

Palavras-chave: *API*, *subscriber*, *publisher*, *Android*, conteúdos georreferenciados e base de dados.

Abstract

Currently mobile applications are indispensable for the mobile user. According to recent studies, the revenues generated by its use resulted in approximately $ 20 billion only in 2013. With these figures in mind, it is estimated that in the near future these numbers will increase, therefore becomes imperative to address a scientific study on this growing market.

Adding to the numerous applications for the *Android* mobile system, there are applications of geo-referenced services that will be addressed here in greater detail, like consumption and provision of these services in mobile clients.

With considered several solutions is possible with some improvements achieve specific benefits to a greater number of users. One of the main issues is the capability of different sources of information (several publishers) to publish georeferenced contents with the use of a documented and opened Application Programing Interface (*API*). The consumption of those contents is done on an Android mobile platform (subscribers), where different publishers can publish their contents according to the user's position.

This publication will be facilitated by the creation of databases for the storage of all relevant information from publishers and subscribers.

The validation of the methods developed was to test a variety of publications, including text, images and charts in the search and consumption on the Android mobile platform. The second phase was the publication of real-time chart on the mobile platform.

Keywords: *API*, *subscriber*, *publisher*, *Android*, georeferenced contents and database.

Índice

Lista de figuras

Lista de tabelas

Siglas e acrónimos

Lista de siglas e acrónimos mais usados nesta dissertação.

UNL	*Universidade Nova de Lisboa*
FCT	*Faculdade de Ciências e Tecnologia*
API	*Application Programing Interface*
OS	*Operating Systems*
LBS	*Location Based Services*
AP	*Access Point*
SIENA	*Scalable Internet Event Notification Architecture*
JEDI	*Java Event-based Distributed Infrastructure*
PADRES	*Principles of Applications of Distributed Event-based Systems*
REDS	*Reconfigurable Dispatching System*
XSIENA	*eXtended Scalable Internet Event Notification Architecture*
STEAM	*Scalable Timed Events and Mobility*
MANET	*Mobile Ad-hoc Networks*
EBS	*Event Broker System*
EB	*Event Broker*
ES	*Event Sources*
ED	*Event Displayers*
N	*Número total de subscrições*
TMAS	*Targeted Mobile Advertising System*
LBMMPS	*Location Based Mobile Multimedia Pusher System*
UI	*User Interfaces*

HTTP	*HyperText Transfer Protocol*
JSON	*JavaScript Object Notation*
ADT	*Android Development Tools*
DER	*Diagrama de Entidades e Relações*

1. Introdução

O projecto desenvolvido no âmbito desta dissertação enquadra-se na área da publicação e pesquisa de conteúdos georreferenciados. Neste primeiro capítulo é dado a conhecer alguns números sobre esta área e qual o problema que se pretende abordar. Para além disso, também são definidos os objectivos a alcançar relativos ao problema que se pretende solucionar, bem como as contribuições gerais da dissertação e sua estrutura.

1.1. Motivação

O crescimento da popularidade de dispositivos móveis está a mudar permanentemente o funcionamento da Internet para os utilizadores. Aparelhos como os *smartphones* ou *tablets* estão a começar a substituir o computador convencional como meios primários de interacção com a tecnologia da informação e recursos *web*, prevendo-se que mais de ¼ da população mundial use *Smartphones* em 2015 [1].

Este crescente consumo, leva a um aumento exponencial das receitas envolvidas com este sector. A Figura 1-1 revela a sua evolução desde 2011 até ao final de 2013, expressando ainda as perspectivas para os restantes anos seguintes até 2017 onde se espera receitas superiores a 70 biliões de dólares [2].

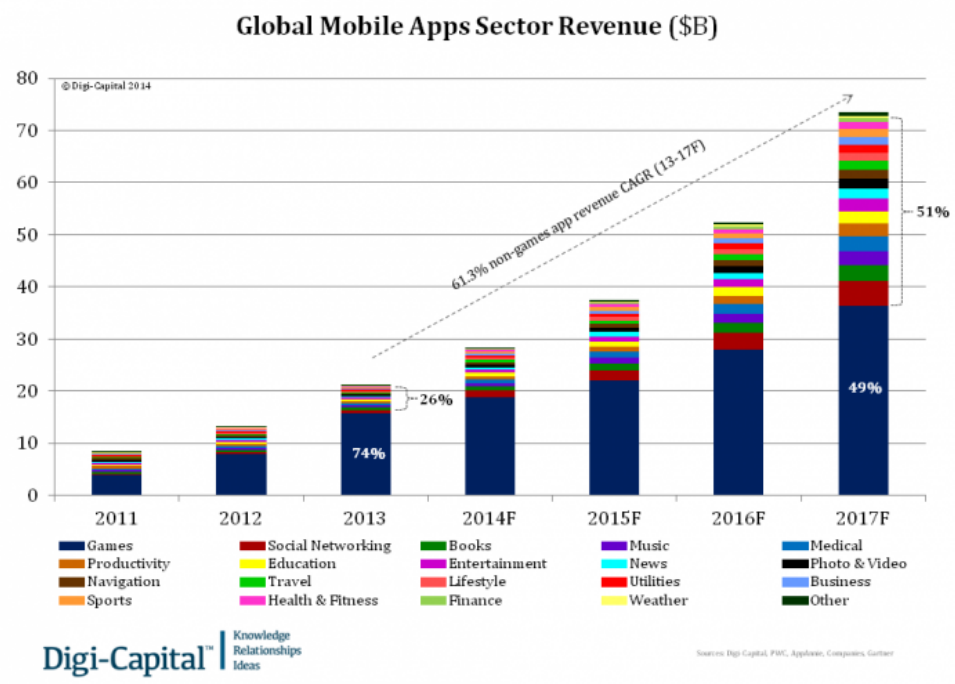


Figura 1-1 Gráfico que exibe o valor das receitas do sector global das Aplicações Móveis em função do ano [2].

O aumento de tanto de receitas como de utilizadores coloca uma enorme pressão para uma constante actualização e criação de *Application Programing Interfaces* (*API*) e aplicações móveis, tornando-se imperativo a modernização e elaboração de conjuntos de rotinas, protocolos e ferramentas para implementação de aplicações *software, APIs* [3].

Existem vários *Operating Systems* (*OS*) para os diversos dispositivos móveis existentes no mercado, destacando-se 3 líderes de mercado, o sistema *Android* do *Google*, o *iOperating Systems* (*iOS)* da *Apple* e o *Windows mobile* da *Microsoft*. Porém, o mercado entre estes líderes não é balanceado, pois mais de 80% do *marketshare* pertence ao *OS Android*, deixando a concorrência muito aquém[4]–[6].

As várias aplicações existentes no mercado diferenciam-se no seu tipo, onde cada uma delas tem as suas vantagens e desvantagens descritas na seguinte Tabela 1 [7]–[9].

Tabela 1 - Tipos de aplicações para os *OS* dos dispositivos móveis[7]–[9].

	Definição	*Vantagens*	*Desvantagens*
Native Apps	**Criado especificamente para as várias necessidades dos vários OS**	▪ Funcionam sem conexão à Internet; ▪ Velocidade, performance e interface com o usuário optimizado.	▪ Aplicação específica para cada *OS;* ▪ Leva mais tempo a desenvolver e implementar; ▪ Custos mais elevados de desenvolvimento.
Web-based Apps	**Websites construídos utilizando HTML, projectados para telas menores**	• Distribuição mais simples (não uso de *iTunes* e *Google Play*); • Funciona em qualquer dispositivo com um navegador; • Menores custos de implementação.	• Desempenho mais lento; • Conexão com a Internet necessária.
Hybrid Apps	**Correm no dispositivo, mas são escritas em tecnologias web (HTML5, CSS e JavaScript)**	• Armazena o conteúdo, por isso funciona *off-line* até um ponto; • *Download* nas lojas dos aplicativos; • Maior facilidade de implementação e menor custo que *Native Apps.*	• Não funciona tão bem quanto os *Native Apps;* • Desempenho *off-line* pode ser inconsistente; • Construído para um *OS* específico.

1.2. Objectivos

Com vista ao desenvolvimento exponencial e obtenção de *one piece of the pie,* o objectivo principal desta dissertação é a criação de uma *API* que facilite a disponibilização de serviços georreferenciados para clientes *mobile.* Para tal, deve-se desenvolver uma plataforma móvel (*Android*) onde vários servidores possam publicar conteúdos georreferenciados, não sendo estes definidos à partida para que a plataforma possa aceitar diversas fontes de informação a partir de vários *publishers.* A existência de um serviço de gestão centralizada de *publishers* é essencial para o correcto funcionamento da comunicação.

A aplicação *mobile* permitirá ao *subscriber*:

- Consumir os diversos tipos de publicações de dados, nomeadamente texto, imagens e a produção de gráficos interactivos com o utilizador;
- Pesquisar os pontos de interesse em torno do utilizador, a partir de um raio escolhido pelo próprio e alertá-lo quando à distância, podendo ou não visualizar no mapa os diferentes pontos;
- Pesquisar os pontos de interesse a partir do seu nome, independentemente da distância a que se encontra o utilizador, podendo ou não visualizar no mapa os diferentes pontos;
- A autenticação nos serviços que o exijam;

A *API* de serviço da aplicação permitirá ao *publisher*:

- A publicação de diversos tipos de conteúdos de dados, como texto, imagem ou dados de forma a produzir gráficos interactivos para serem consumidos pela aplicação *mobile (subscriber),* sendo possível o seu armazenamento num servidor *web*;
- Visualizar as propriedades dos seus conteúdos e personalizar o seu perfil;
- A administração, sem uso abusivo da sua conta, por parte de um *publisher* administrador.

O *Web Server* conterá:

- As estruturas necessárias para o armazenamento dos dados dos *publishers* e *subcribers* e seus conteúdos.

1.3. Contribuições Gerais

A contribuição desta dissertação é a criação de uma aplicação móvel e *API* que permitirá ao utilizador publicar, pesquisar e consumir os mais variados géneros de conteúdos em tempo real a partir da sua posição.

1.4. Estrutura da dissertação

- Capítulo 2: Estado da Arte

Abordar-se-á o funcionamento de modelos *publisher* e *subscriber*, bem como a tipologia *broker*, a aquisição de dados a partir da localização do utilizador, o funcionamento de *OS Android* e protocolos de transmissão de dados usados pela proposta.

- Capítulo 3: Proposta (Modelo Conceptual)

A explicação da solução proposta, olhando primariamente para o modelo de interacção entre entidades, em seguida para as estruturas das diferentes entidades, finalizando com o seu funcionamento.

- Capítulo 4: Validação

São apresentados os métodos utilizados para a validação e fiabilidade da solução proposta.

- Capítulo 5: Conclusão e Perspectivas

É feita uma reflexão sobre o trabalho desenvolvido, focando os cenários de aplicação e indicação de pontos de interesse para pesquisa futura.

2. Estado da Arte

Neste capítulo abordar-se-á primeiramente o funcionamento de modelos *Publisher – Subscriber* para um sistema distribuído e suas relevantes categorias, arquitecturas e sistemas. Seguidamente, informar-se-á sobre a utilização da topologia *Broker* em modelos *Publisher – Subscriber*. Em seguida, explicar-se-á o tema da aquisição de dados a partir da localização, *Location Based Services* (LBS), juntamente com o seu uso no *Operating System* (*OS*) *Android*. Por fim, informar-se-á sobre os protocolos de transmissão de dados usados na implementação.

Para além disso, será feito um levantamento das principais aplicações móveis concorrentes e outros conhecimentos computacionais importantes para a implementação e validação da proposta.

2.1. Categoria de modelos *Publisher – Subscriber* para um sistema distribuído

Um sistema com o modelo *publish/subscribe* é uma estratégia para estabelecer uma comunicação entre os provedores (*publishers*) e os consumidores (*subscribers*) de informação num sistema distribuído. Estes diferem nas suas características fundamentais, decompondo-se em duas categorias gerais: sistemas *Subject-based* e *Content-based* [10].

2.1.1. Subject-Based

Neste modelo, os eventos são marcados baseados num conjunto fixo de tópicos ou *subjects*. A subscrição é feita com o objectivo de o utilizador receber eventos associados ao tópico [11]–[17].

A Figura 2-1 esquematiza como é que a comunicação entre as entidades é feita. Os vários *publishers* inserem conteúdos com os respectivos *subjects*. Já o *subscriber* pede a subscrição a um *subject*, o sistema recolherá os vários dados sobre o mesmo e enviará a informação para ser consumida pelo *subscriber*. Mais tarde, se este desejar receber mais informação sobre algum dado recebido em específico poderá pedir ao sistema.

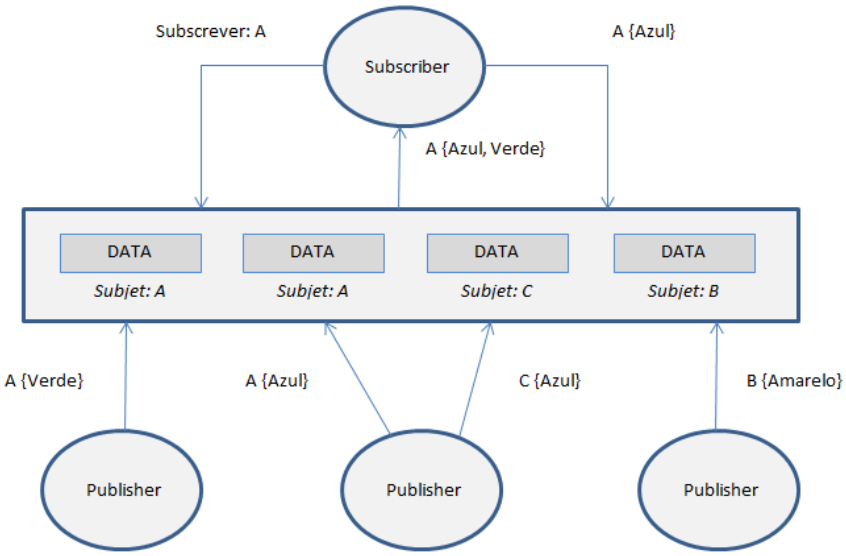


Figura 2-1 Esquema de um sistema *Subject-based*, com um exemplo de uma subscrição de um *subscriber* no *subject* A.

2.1.2. *Content-Based*

Por outro lado, num sistema *Content-based*, os *subscribers* podem redefinir as suas subscrições escolhendo critérios de múltiplas dimensões sem precisarem de um tópico/*subject* definido [11]–[17].

A Figura 2-2 esquematiza como é que a comunicação entre as entidades é feita. Neste caso, não é necessário perder tempo na subscrição de um *subject* e, em vez disso, o *subscriber* pede logo uma lista de condições para os seus dados {Azul}. O sistema seleccionará a DATA para os dados relevantes, nos vários *subjects* e responderá ao pedido.

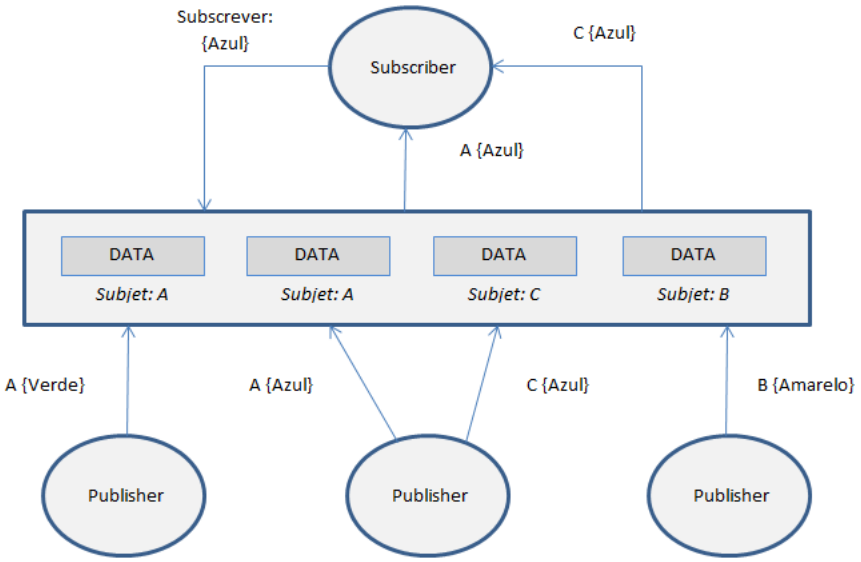


Figura 2-2 Esquema de um sistema *Content-based,* com um exemplo de uma subscrição de um *subscriber* nos *subjects* da cor Azul.

Porém, este tratamento dos dados leva a um encargo no sistema adjacente para coincidir as mensagens com as subscrições, podendo este número ser muito superior ao dos *subjects*, que devem ser geridos num sistema *Subject-based* [10].

2.2. Arquitectura de modelos *Publisher – Subscriber* para um sistema distribuído.

A comunicação entre os vários componentes de *software* não é mais do que um sistema de eventos, onde estes podem participar como servidores, clientes ou ambos. Assim, a arquitectura do sistema pode ser classificada em duas categorias: *client-server* ou *peer-to-peer* [10].

2.2.1. Client-Server

Neste modelo (ver Figura 2-3), uma das componentes opera como servidor ou cliente de eventos. Os servidores estão incumbidos da recepção, armazenamento e encaminhamento desses pedidos, comunicando uns com os outros para a eficiência do sistema. Os clientes funcionam como *publishers*, *subscribers* ou ambos [10], [15].

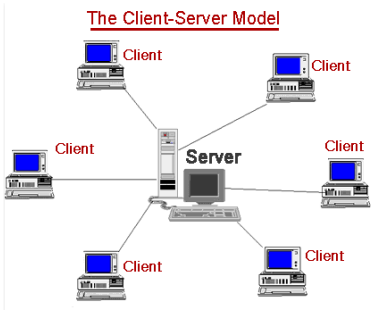


Figura 2-3 Ilustração sobre um modelo *Client-Server* [18].

Na Figura 2-4 estão ilustradas as diferentes topologias gerais dentro desde modelo.

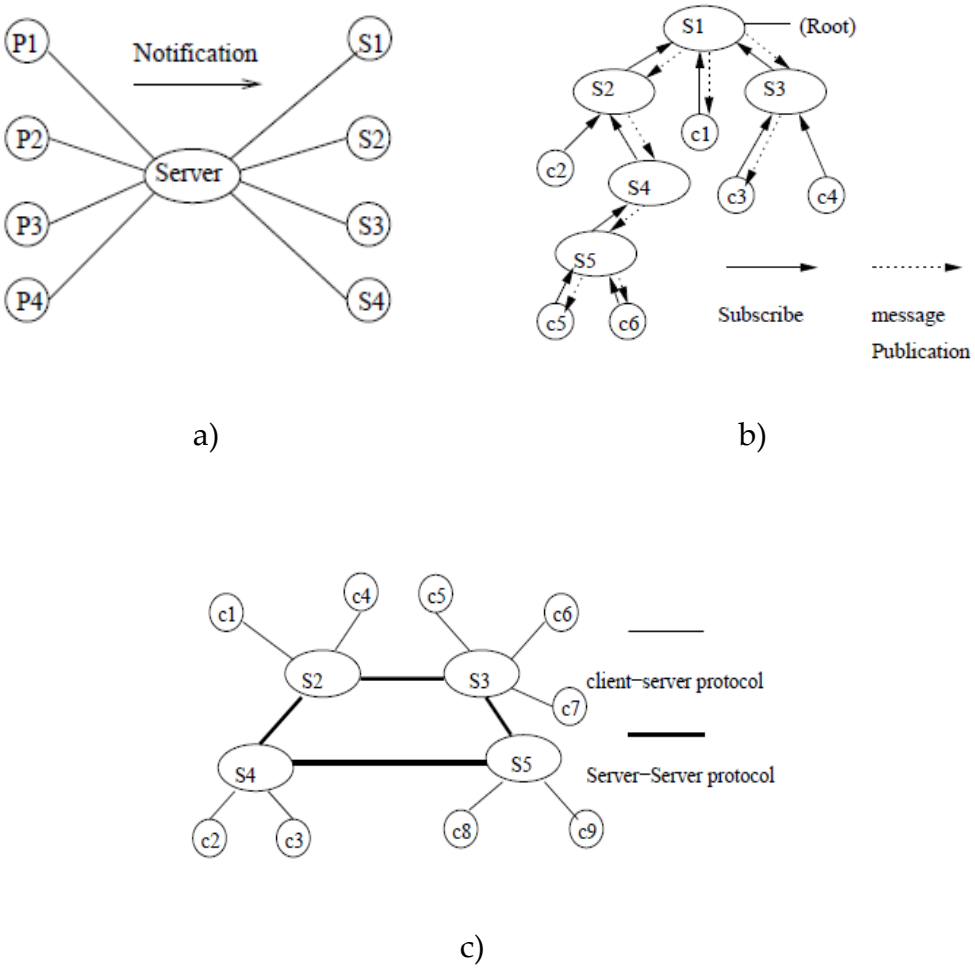


Figura 2-4 Topologias *Client-Server*: a) Estrela (centrada no servidor); b) Hierárquica de Servidor; c) Anel ; P – *Publisher*, S – *Subscriber* em a); S – Servidor em b) e c) ;e C – Cliente [10].

A topologia *a)* da Figura 2-4 é centrada no servidor, dependendo de um único servidor para intermediar os *publishers* e *subscribers*, podendo estes receber eventos de vários *publishers*.

Este intermediário pode ser conhecido como *Broker*, e será um conceito debatido neste capítulo, nomeadamente no subcapítulo 2.4 [11], [16], [19].

Na topologia *b)* da Figura 2-4, tal como o nome o sugere, existe uma relação hierarquia entre os servidores, podendo os clientes agir como *publishers* ou *subscribers*.

O protocolo de comunicação é idêntico entre servidor-servidor e servidor-cliente, não distinguindo as mensagens trocadas entre eles. O servidor principal irá receber os eventos e encaminha-os para as subáreas destinadas[10], [12], [14], [17].

Na topologia *c)* da Figura 2-4, os servidores funcionam entre eles numa relação de *peer-to-peer*, fazendo um anel, onde a comunicação é bidireccional. A comunicação entre cliente-servidor é diferente da servidor-servidor, pois a última mantem-se actualizada constantemente, enquanto a outra só é criada quando um cliente gera um pedido de subscrição ou publicação. O servidor pode funcionar como AP (*Access Point*) ou *router* [10], [12], [14], [17].

2.2.2. Peer-to-peer

No modelo *Peer-to-peer* existe uma equabilidade entre os nós, onde cada um pode funcionar como *publisher*, *subscriber*, raiz ou nó interno de uma *multicast tree* [10], [15].

2.3. Sistema de modelos *Publisher – Subscriber* para um sistema distribuído

Existem vários sistemas que implementam protocolos e metodologias *publisher/subscriber*.

Os sistemas *Gryphon*, *SIENA*, *Elvin*, *JEDI*, *XSIENA*, *Rebeca* e *PADRES* são bons exemplos de soluções de subscrição *content-based*, suportando-se em infra-estruturas baseadas em *brokers* e construídas acima da *network layer*, onde o encaminhamento é feito com a utilização de filtros agregação, dinâmicos ou de balanceamento de tráfego [10], [16].

Os sistemas *Echo e JEcho* utilizam o mesmo tipo de infra-estruturas e filtros de encaminhamento, com a adição do filtro da mobilidade para o *JEcho*, mas diferenciam-se na sua forma de subscrição. No *Echo* esta é feita através de um canal, enquanto no *JEcho* é feita através de moduladores [16].

Os sistemas *Green* e *REDS* utilizam infra-estruturas de módulos conectados estruturados ou não para darem suporte ao encaminhamento em módulos conectados de grandes áreas, redes *ad hoc* e reconfiguração da topologia para o caso *REDS*. As subscrições são feitas através de variadas conexões e para o caso dos *REDS* pode ser da forma: pedido-resposta [16].

O sistema *Fuego* é um modelo de subscrição *content-based,* onde as infra-estruturas variam consoante o *cluster* federado e o encaminhamento é feito ao encontro da conexão entre *publisher* e *subscriber* por um mediador (*rendezvous*) [16].

No sistema STEAM é utilizado um modelo de subscrição *subject-based* e de proximidade. O encaminhamento é feito com vista à proximidade entre os nós, numa infra-estrutura onde os grupos são atribuídos a áreas geográficas específicas [16].

Para ajudar a compreender melhor estes sistemas, existe uma Tabela 32 resumo com estas informações nos anexos.

2.4. Topologia *Broker*

Um sistema *publisher/subscriber* liga vários fornecedores de informação e consumidores, oferecendo eventos a partir de fontes para os utilizadores interessados. Quando um evento é gerado e publicado para o sistema, uma infra-estrutura é responsável pela verificação do evento contra todas as assinaturas atuais e por entregá-lo de forma eficiente e confiável para todos os utilizadores cujas subscrições vão coincidir com o evento [11], [13], [17], [19]–[25].

A Figura 2-5 esquematiza este conceito generalizado, onde existe um ou mais *Event Source* (*ES*), um *Event Broker System* (*EBS*) e um ou mais *Event Displayer* (*ED*). O *ES* gera os eventos e publica-os no *EBS*, e consoante o conjunto de subscrições feitas, o *EBS* irá encaminhá-los para os respectivos *EDs* [19].

O *EBS* é um serviço de notificação e entrega eficiente de eventos, para além de armazenar e gerir as assinaturas (*subscribers*). Este funciona como um mediador neutro entre as entidades, os *publishers*, na qualidade de produtores de eventos e os *subscribers*, na qualidade de consumidores [17].

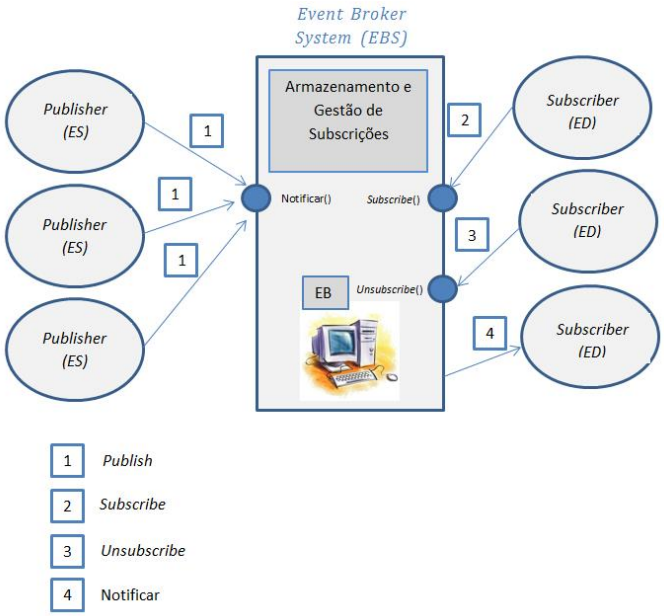


Figura 2-5 Um modelo simples de um sistema *Publisher/Subscriber* com a utilização de um intermediário, *Event Broker System (EBS)*. *Event Source* (ES); *Event Displayer* (ED); *Event Broker* (EB).

A utilização das diferentes categorias explicadas no subcapítulo 2.1 permite diferentes técnicas de recolhas de eventos. Existem vários géneros de aplicações desde conceito, *Event Broker (EB)*. Os seguintes subcapítulos abordaram as mais relevantes para esta dissertação: modelo centralizado, modelo distribuído e modelo replicado.

2.4.1. Modelo centralizado

Neste modelo, um servidor faz toda a correspondência e filtragem. Este servidor é localizado num computador, pois um dispositivo móvel não possui a mesma capacidade de alocação e processamento de dados e os *publishers* podem ser anónimos e não permitir aos utilizadores o armazenamento das suas subscrições. O *EB* deverá estar localizado numa rede fixa, para evitar uma desconexão e consequente paralisação do sistema [19].

A Figura 2-6 esquematiza dois modelos centralizados para interacção *publisher/subscriber*, baseada numa categoria *Content-Based*. O *EB*, após verificar os critérios de subscrição (*c*), encaminhará o evento para os *users* (*u*).

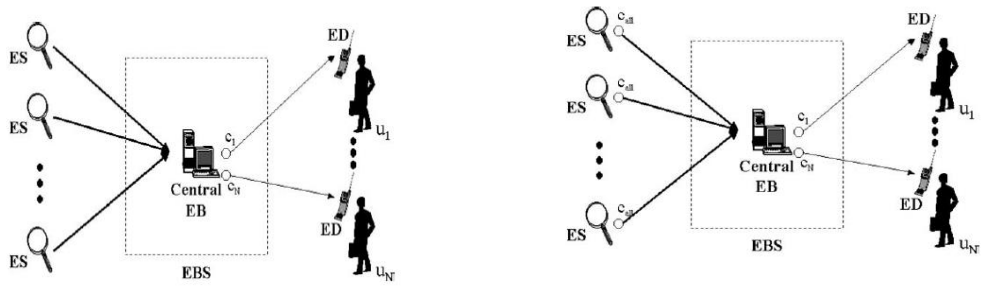


Figura 2-6 Esquemático de modelos centralizados: a) Arquitectura centralizada; b) Arquitectura centralizada com *quenching*. *c* – critério de subscrição e *u* – user [19].

A utilização de *quenching* num sistema *Content-Based* permite a verificação dos critérios de subscrição (*c*), ou seja, averigua se algum utilizador final está interessado no evento ou não. Se alguma subscrição em *EB* não corresponder ao evento desejado ($C_{all}(e) = false$), o evento será descartado na origem(*ES*), reduzindo o tráfego e o processamento do *EB* central [19].

A Figura 2-7 mostra a importância de um *EB* para gerir as entregas de pacotes numa rede móvel. No início a comunicação é directa entre receptor e emissor, não se encaixando neste conceito descrito. Porém, quando o receptor se move para fora da rede onde se encontra o emissor, a importância de um *broker* com a informação permitirá ao receptor continuar a receber a informação [23].

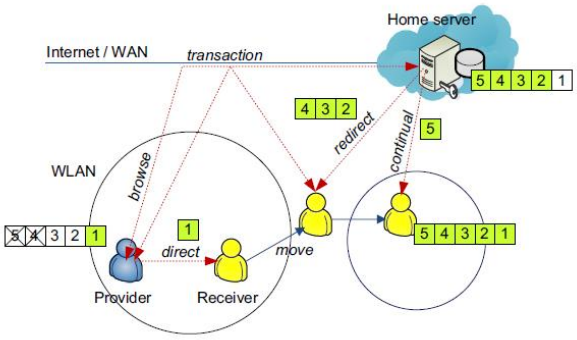


Figura 2-7 Processo de envio de envio de pacotes entre um *publisher* e *subscriber* móvel, com a utilização de um *broker* centralizado [23].

2.4.2. Modelo Distribuído

Neste modelo, a centralidade de um servidor único é dissipada. Os vários *EBs* estão distribuídos pela rede, podendo operar de várias maneiras como mostra a Figura 2-8.

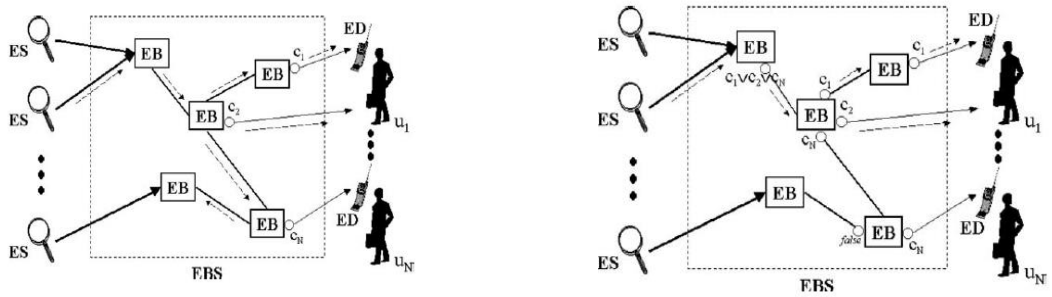


Figura 2-8 Exemplos de modelos distribuídos: a) *Broadcast* e b) *Multicast*. As linhas a tracejado são caminhos de um evento exemplo para satisfazer c_1 e c_2 [19].

Em ambos os modelos pode-se verificar que o *EBS* é constituído por múltiplos *EBs*, sendo que cada um deles é responsável por uma proporção do número total de subscrições (*N*). Um *ES* publica um novo evento num dos *EBs* (normalmente é o mais próximo), responsabilizando este pelo encaminhamento do evento para os outros *EBs* do *EBS*. O *EB* onde o *ED* está ligado fará a entrega final do evento [19].

O encaminhamento no *Broadcast* é feito através de um *flush* no sistema com o evento, enquanto no *Multicast* é feito através de um algoritmo chamado *Distributed Multicast*. Neste caso, o evento só é passado para os *EBs* que pertencem ao *Forwarding Tree* que corresponde à subscrição desejada. Criando, um "*pruned*" para os caminhos indesejados [19], [26].

Para a adaptação móvel, é necessário ter em conta que os *EDs* e os *ESs* movimentam-se, podendo-se desconectar e conectar a outro *EB* diferente. Quando isto acontece, é necessário reajustar a *routing tree* para o encaminhamento directo de eventos relevantes. Não só isso, mas também é necessário o *EB* armazenar as subscrições de eventos pendentes do *ED* antes da desconexão para as enviar assim que o *ED* voltar a conectar-se [19], [26].

Em alternativa, o *ED* pode carregar informação sobre as suas subscrições e carregá-la no novo *EB* quando reconecta. Isto permite ao *ED* receber novos eventos, mesmo que o antigo *EB* esteja temporariamente em baixo. A desvantagem encontra-se na possibilidade de vários *EBs* poderem monitorizar as mesmas subscrições do mesmo *ED*, podendo ser combatido com informações sobre as conexões passadas, tanto *IDs* dos *EBs* como temporais [19].

2.4.3. Modelo Replicado

Este modelo é utilizado para aumentar a viabilidade e fiabilidade do sistema em caso de avarias. A Figura 2-9 mostra a implementação de um *EBS* replicado.

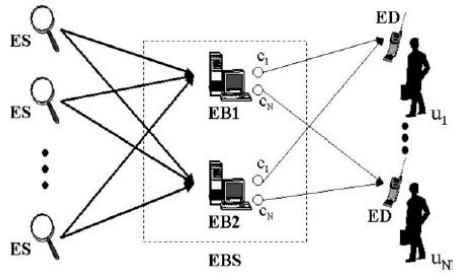


Figura 2-9 Esquemático de modelo replicado [19].

Os vários *EBs* monitorizam simultaneamente as subscrições de cada utilizador, havendo menor probabilidade do utilizador perder um evento, pois os dois fluxos de eventos gerados por *EB1* e *EB2* fundir-se-ão em *ED*. Este modelo leva a problemas de ordem das mensagens, consistência e plenitude [19]. A Tabela 2 mostra a influência de vários filtros no *ED* para redução destes problemas.

Tabela 2 Propriedades satisfeitas para um sistema replicado *Publish/Subscribe* dependendo de vários algoritmos de filtros *ED* [19].

ED filtering	Orderedness	Consistency	Completeness
No filtering	X	X	√
Duplicate removal	X	√	√
Out-of-order and duplicate removal	√	√	X

Cada um dos algoritmos aplicados não garante a resolução de todos os problemas, mas ajuda a melhorar o sistema.

2.5. *Location Based Services* (*LBS*)

Os *LBS* definem-se como uma classe geral de serviços de nível programático, onde o controlo de recursos é feito a partir de dados posicionais [27]–[35]. A Figura 2-10 ilustra este conceito, centrando as ideias a partir do *LBS* ao centro, onde o primeiro anel (verde) representa conceitos envolvidos no estudo de *LBS*. O segundo (azul) demonstra as tecnologias que estão associadas com os conceitos subjacentes. O externo (laranja) ilustra como os utilizadores acedem ao *LBS*, através *service providers* e aplicações que podem utilizar uma interface de *Application Programing Interface* (*API*) [32].

Figura 2-10 Diagrama sobre o conceito LBS[32].

O conhecimento da localização do utilizador final é importante para fornecer conteúdos e informações relevantes e envolventes. Para além disso, para os operadores de redes móveis, estes serviços representam um fluxo adicional de receitas que podem ser gerados a partir de seus investimentos em infra-estrutura fixa[34].

O conhecimento da posição geográfica beneficia mutuamente as três entidades envolvidas: o utilizador no seu dispositivo móvel, o provedor da tecnologia de posicionamento (normalmente um telemóvel ou uma rede sem fios) e o *service provider* [35].

Este benefício é feito através da personalização destes serviços, sendo atribuído ao cliente a sua gestão, pois este raramente deseja receber toda a informação da área relevante [33]. A implementação *Tiled Feeds* [33] é um exemplo desta personalização onde as diferentes *views* são personalizadas tanto pelos clientes *web* como nos *mobile* sem a utilização de um *adapter* específico para cada *LBS,* integração horizontal.

O uso deste conhecimento permite a publicação de publicidade relevante para o utilizador. Pois, ao saber a informação personalizada sobre um cliente, é possível fazer um marketing mais efectivo para a localização do utilizador. Como por exemplo, fazer anúncios sobre lojas de artigos desportivos para um utilizador que costuma ir a eventos desportivos correntemente[29], [34].

2.5.1. Aplicações que utilizam esta tecnologia

A *Targeted Mobile Advertising System* (*TMAS*) [29], permite providenciar tanto os publicitários como os consumidores com publicidade *context-aware*. A Figura 2-11 demonstra a estrutura de um sistema com a implementação. Onde a partir dos dados tanto dos consumidores como dos publicitários, pode-se fazer um marketing direccionado, devido ao cruzamento de interesses.

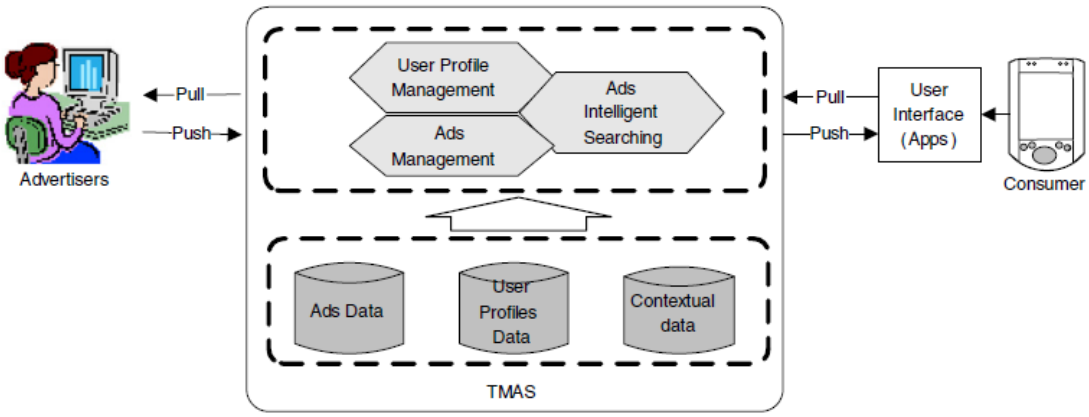


Figura 2-11 Esquema do funcionamento de um sistema *TMAS* [29].

O *Location Based Mobile Multimedia Pusher System (LBMMPS)* [36] utiliza as coordenadas de GPS para introduzir eventos no *Google Maps*, onde os utilizadores podem indicar a localização desse evento ocorrido na interface do mapa. A Figura 2-12 esquematiza a arquitectura do sistema.

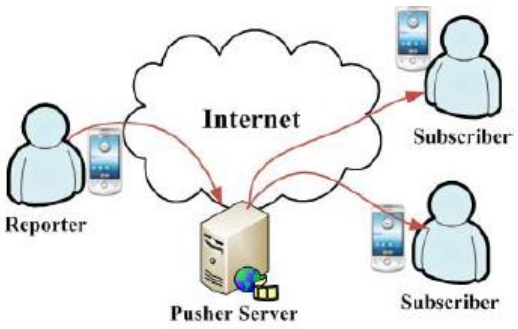


Figura 2-12 Arquitectura de um sistema LBMMPS [36].

Quando um *reporter* captura fotos ou filmes, o dispositivo móvel *android* envia instantaneamente o conteúdo e as coordenadas GPS para o *pusher server*, para este reencaminhar o pedido para o *subscriber* [36].

2.6. Funcionamento de uma aplicação *Android*

Uma aplicação desenvolvida para o *Android Operating System (OS)* tem por base a linguagem programática específica *Java*, se for do tipo *Native App*. Este *OS* foi criado em 2008, vindo a evoluir até às versões mais recentes 4.4 (*KitKat*) e 5.0 (*Lollipop*), sendo esta última a entrar no mercado [37]–[48].

A instalação e distribuição das aplicações é feita através dos seus ficheiros *.apk*, tornado-se a *Google Play* como loja *on-line* destes ficheiros. Porém, antes da aplicação ficar disponível neste domínio são necessárias várias licenças[37]–[48].

As várias componentes de uma aplicação *Android* [37]–[48] são:

- *Activities*: vistas especiais com suas próprias *User Interfaces* (*UI*). O seu ciclo de vida está exemplificado nos anexos na Figura 5-1;
- *Services*: Representam uma tarefa *background* da aplicação;
- *Content Provider*: tem como objectivo principal partilhar as fontes de dados com outros componentes;
- *BroadCast Receiver:* são actividades de outro sistema ou eventos da aplicação.

2.6.1. Location Base Services (LBS)

O *OS Android* permite o uso de bibliotecas para obtenção da posição actual, e uso de serviços a partir da posição do utilizador. A classe *Location Manager*[49] é um dos exemplos para obtenção de posição actual, fazer rastreamento dos movimentos e colocar alertas de proximidade quando um utilizador entra numa determinada área. Para além disso, utiliza uma localização precisa de *GPS* ou localizações baseadas no *WiFi* ou *BS,* dependendo das permissões e da precisão da posição [43], [49], [50].

O uso de *Google Play Services*[51] permite a visualização de coordenadas no *Google maps*. A partir da criação de uma *Google API Key*[52] para o projecto, consegue-se obter uma vasta gama de serviços Google, entre eles o uso do mapa[43], [51], [52].

2.6.2. *Content Provider*: *Android SQLite*

O *OS Android* permite o uso do *Content Provider SQLite*[53]. Este provedor não só suporta recursos do banco de dados relacionais padrão, como a sintaxe *SQL,* transacções e instruções preparadas, mas também tem tipos de dados próprios muito similares ao *Java*[39], [53]–[55].

2.7. Protocolos de transmissão de dados entre Servidor e Aplicação

Existem vários protocolos de transmissão de dados nas diversas camadas da Internet, porém como esta implementação se encontrará na camada da aplicação, ir-se-á olhar com mais detalhe para dois protocolos: o *JavaScript Object Notation* (*JSON)* e o *HyperText Transfer Protocol (HTTP).*

2.7.1. *JavaScript Object Notation* (*JSON*)

JSON para além de ser um formato de intercâmbio de dados leve é também um formato de texto que é completamente independente da linguagem, mas usa convenções que são familiares aos programadores da família C. A sua construção é composta por duas estruturas: um objecto e uma *array* [56]–[58].

Um objecto é conjunto desordenado de pares nome / valor. Um exemplo da sua estrutura está definido no seguinte conjunto: { nome : John , idade : 25 } [56]–[58].

Uma *array* é uma colecção ordenada de valores. Um exemplo da sua estrutura está definido no seguinte conjunto de tipos: [1(numérico) , Rui(texto) , objecto (objecto) , true (booleano) , null] [56]–[58].

2.7.2. HyperText Transfer Protocol (HTTP)

HTTP é um protocolo de aplicação de pedidos distribuídos, colaborativos e de sistemas de informação hipermédia, para além de ser a base de comunicação de dados para o *World Wide Web*[59]–[63].

O *HyperText* é um texto estruturado que utiliza ligações lógicas (hiperligações) entre os nós que contém texto. O HTTP é um protocolo para troca ou transferência de *HiperText* [59]–[63].

Existem vários métodos *request*, a serem executados no recurso identificado pelo *Request-URI*. A Tabela 3 mostra alguns dos métodos de *request* suportados no HTTP/1.1:

Tabela 3 – Métodos *requests* HTTP e suas descrições [59]–[63].

Método	Descrição
GET	É usado para recuperar informações do servidor usando um determinado *URI*. Este pedido só recupera os dados e não deve ter nenhum outro efeito sobre eles.
HEAD	Idêntico ao *GET*, mas este transfere só a *status line* e a *header section*.
POST	É utilizada para enviar dados para o servidor, usando formulários *HTML*.
PUT	Substituí todas as *current representations* do recurso destino com o conteúdo enviado.
DELETE	Remove todas as *current representations* do recurso destino dado pelo *URI*.
CONNECT	Estabelece um túnel para o servidor, identificado por um determinado *URI*.
OPTIONS	Descreve as opções de comunicação para o recurso destino.
TRACE	Executa um *loop* de mensagem de volta junto com o caminho para o recuso destino.

3. Proposta (Modelo Conceptual)

Neste capítulo é descrito o modelo de interacção entre entidades e tecnologias utilizadas, bem como as várias estruturas do sistema, desde a *API* e base de dados até à da aplicação móvel (*Android*). Por fim, explica-se o modelo de funcionamento do sistema, relacionando os protocolos de comunicação e as várias estruturas anteriores.

3.1. Modelo de Interacção entre Entidades

A proposta desta dissertação é a criação de uma *API* e aplicação *mobile* de forma a permitir o consumo e criação de conteúdos georreferenciados. Com base nos conhecimentos do capítulo anterior, a estrutura de interacção entre entidades escolhida será a ilustrada na Figura 3-1.

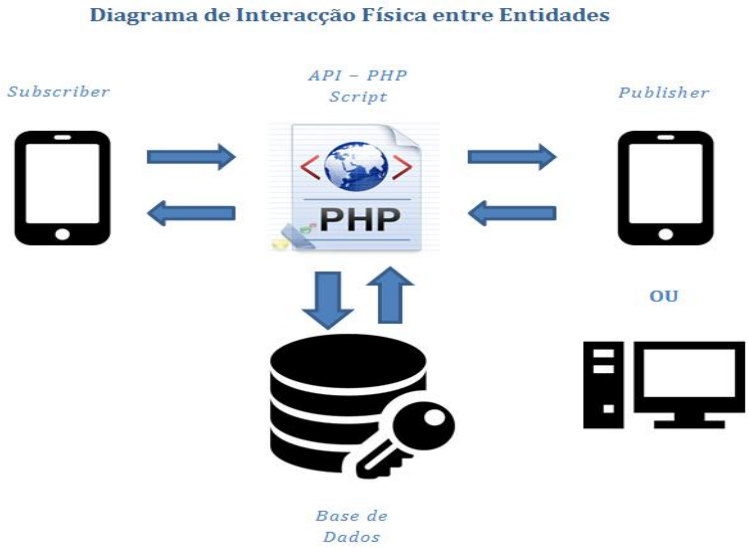


Figura 3-1 Diagrama de Interacção entre as entidades *subscriber*, *publisher*, *API* e base de dados.

O *subscriber*, a partir de uma *Native App* do seu telemóvel, interage com a *API* utilizando o formato de intercâmbio de dados *JavaScript Object Notation* (*JSON*).

A responsabilidade da comunicação com o *web server* está ao cargo da *API*, sendo que tanto o *subscriber* como o *publisher* nunca comunicam directamente com o servidor.

O *publisher*, a partir de uma aplicação *Web-based* que pode ser acedida a partir de um *browser* num dispositivo fixo ou móvel, interage com a *API* utilizando a comunicação *HyperText Transfer Protocol* (*HTTP*) *GET* e *POST request*.

Para o desenvolvimento desta solução poder-se-ia escolher várias tecnologias, porém por escolha pessoal utilizou-se três *softwares* para criação, implementação e gestão deste sistema, o *Eclipse Android Development Tools (ADT)* [64], *WAMP server* [65] e *NetBeans 8.0.1* [66].

3.2. Estrutura da *API*

A estruturação da *API* e das diferentes tabelas para armazenamento de informação na base de dados são fundamentais para o correcto funcionamento do sistema. Esta estrutura deve ser prática, simples e funcional de forma a cumprir todas as tarefas e pedidos de dados do *publisher* e do *subscriber* de forma eficiente e eficaz, para além de facilitar o desenvolvimento de novos programas fornecendo todos os blocos de construção necessários. Com base nos conhecimentos do capítulo 2, a estrutura escolhida para a *API* é a seguinte:

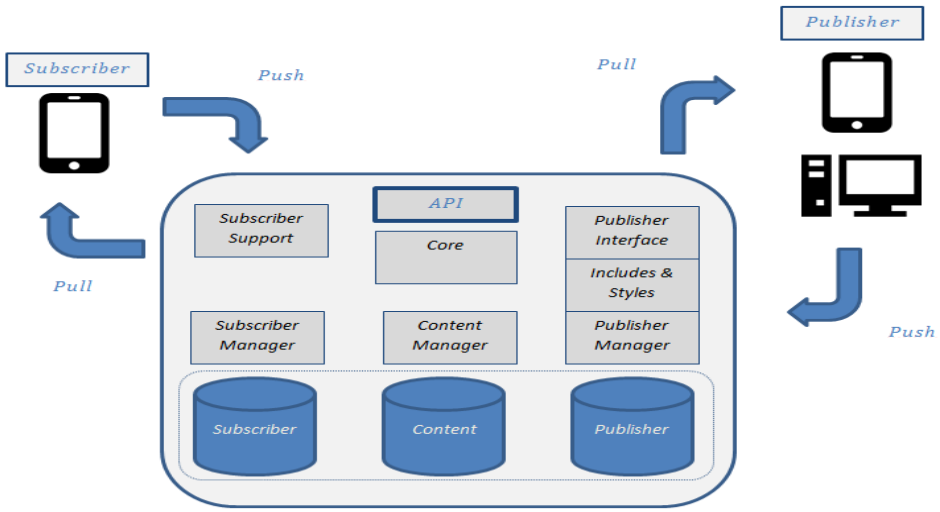


Figura 3-2 Estrutura interna da *API* e base de dados.

A partir da Figura 3-2 pode-se verificar cinco subdivisões no tratamento da informação. A subdivisão de tratamento de dados do *subscriber*, do *publisher*, dos conteúdos, o *core* e o armazenamento dos dados, sendo esta última abordada no subcapítulo 3.3. Nos próximos subcapítulos aprofundar-se-á os conhecimentos sobre cada uma destas subdivisões.

3.2.1. Tratamento de dados do *subscriber*

- *Subscriber Support*: Este módulo é responsável por tratar de toda a comunicação entre a *API* e o *subscriber*. Aqui, é feita a descodificação das várias *tags* do vector *JSON* recebido da parte do *subscriber* e sua posterior codificação para resposta. A informação é tratada e passada para as necessárias verificações para o próximo módulo, o *Subscriber Manager*.

- *Subscriber Manager*: Este módulo é responsável pela criação, inserção e gestão da base de dados relativa ao *subscriber*. A informação já descodificada sofre uma sanitização, e em seguida é utilizada na *query* para comunicar com a base de dados, retornando o resultado para o *Subscriber Support.*

3.2.2. Tratamento de dados do *publisher*

- *Publisher Interface:* Este módulo é responsável por gerir a *interface* e os pedidos do *publisher*. Aqui encontra-se o núcleo da diferenciação das várias páginas, pois é neste módulo que está presente o corpo do *website* responsável pela digitalização e verificação dos dados inseridos. A verificação dos pedidos *HTTP GET* e *POST* é feita neste módulo.

- *Include & Styles:* Este módulo é responsável pela criação do resto do corpo do *website*, nomeadamente o *head*, o *header*, o *footer*, a *navigation*, a *sidebar*, os vários *widgets* e os estilos do *design* do *website*. Este módulo é um auxiliar para o módulo *Publisher Interface*, facilitando a criação de novas páginas *web* pois o módulo auxiliado só tem que se preocupar com o corpo da página.

- *Publisher Manager:* Este módulo é responsável pela criação, inserção e gestão da base de dados relativa ao *publisher*. A informação sofre uma sanitização, e em seguida é utilizada na *query* para comunicar com a base de dados, retornando o resultado para o *Publisher Interface.*

3.2.3. Tratamento de dados dos conteúdos

Nesta subdivisão vão entrar todos módulos já abordados anteriormente, pois o tratamento de dados de um conteúdo está associado ou a um *publisher* ou a um *subscriber*. Sendo que o *publisher* poderá adicionar, visualizar ou eliminar um conteúdo, enquanto o *subscriber* o consumirá.

- *Publisher Interface:* Este módulo será também responsável pela *interface* de alertas sobre o correcto tratamento de dados relativos ao conteúdo. Fazendo as verificações da correcta inserção dos dados e formulários a partir das configurações presentes na subdivisão *core.* Em seguida, passará a informação para o *Content Manager*.

- *Subscriber Support:* Este módulo será também responsável pela *interface* de alertas sobre o correcto tratamento de dados relativos ao conteúdo. Fazendo a descodificação do pedido do conteúdo e suas propriedades, passando a informação para o *Content Manager*, para comunicação com a base de dados do *content*.

- *Content Manager:* Este módulo é responsável pela criação, inserção e gestão da base de dados relativa ao *content*. A informação sofre uma sanitização, e em seguida é utilizada na *query* para comunicar com a base de dados, retornado o resultado para o *Publisher Interface* ou *Subscriber Interface,* dependendo de onde veio o pedido.

3.2.4. Core

- *Core:* Este módulo é responsável pela conexão e credenciais de *login* com a base de dados. Para além disso, contém funções fundamentais para o funcionamento dos outros módulos da *API.*

3.3. Estrutura da base de dados

Para facilitar a visualização da estrutura da base de dados, o Diagrama de Entidades e Relações (*DER)* da Figura 3-3 esquematizará as componentes das diferentes tabelas, bem como as suas relações.

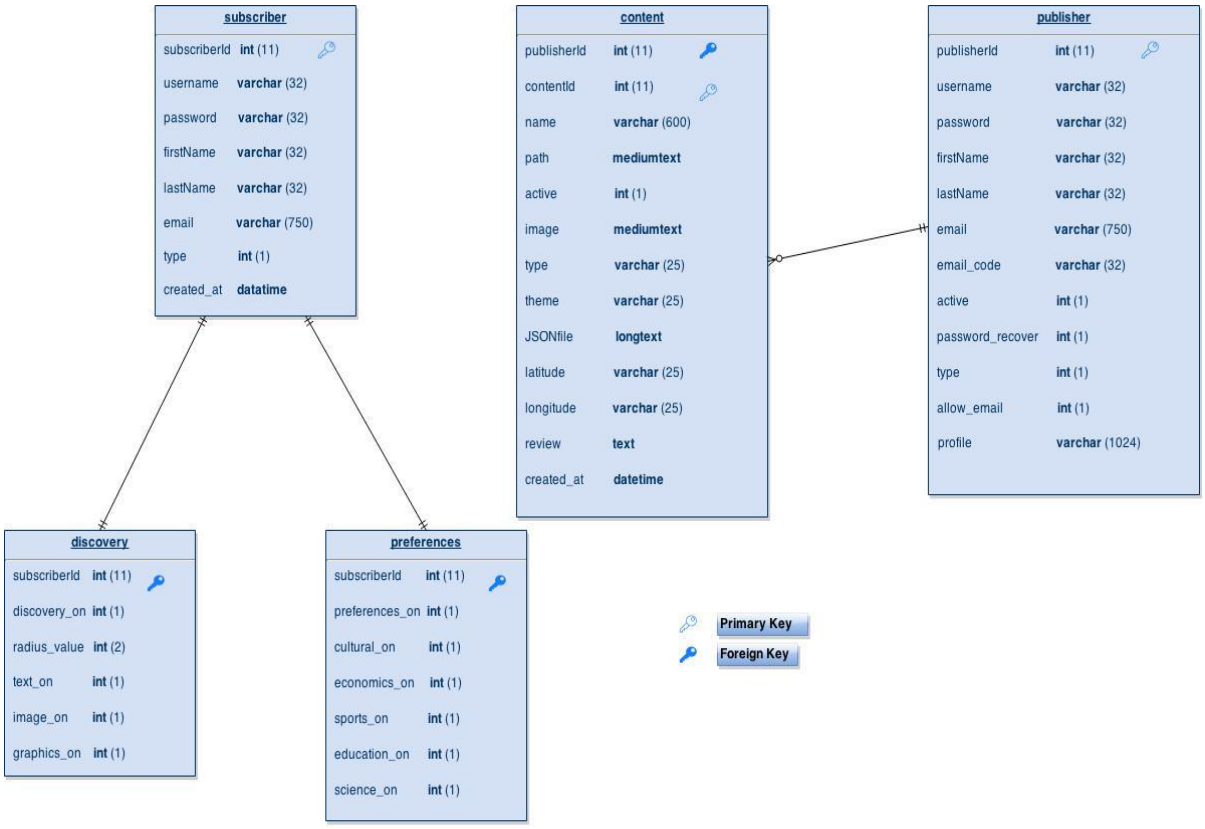


Figura 3-3 Diagrama *DER*, da estrutura das tabelas da base de dados.

A partir da informação do diagrama, pode-se verificar a criação de 5 tabelas para assistir a *API*. A criação e utilização de chave primária (*primary key*) permite a uma coluna ou um grupo de colunas identificar unicamente uma linha (*row*). A chave estrangeira (*foreign key*) é uma coluna ou grupo de colunas em uma tabela cujos valores devem ter valores correspondentes na chave primária de outra tabela adjacente, sendo uma referência a uma chave primária de uma tabela.

A tabela *subscriber* tem como chave primária a coluna *subscriberId*, sendo este valor único para cada *subscriber*, tornando-se uma chave estrangeira para as tabelas subjacentes *discovery* e *preferences*. Estas tabelas foram criadas de forma a reduzir a carga de informação contida na tabela *subscriber*, facilitando até a sua utilização para trabalhos futuros sobre estatísticas de cada utilizador, sem comprometer os seus dados pessoais. Para cada *subscriber* criado, também são criadas as linhas (*rows*) associadas ao mesmo nas tabelas *discovery* e *preferences*, pois estas tabelas encontram-se relacionadas com a tabela *subscriber* numa relação de um para um.

A tabela *publisher* tem como chave primária a coluna *publisherId,* sendo este valor único para cada *publisher*. Esta chave torna-se uma chave estrangeira para a tabela subjacente *content.* Estas duas tabelas relacionam-se uma relação de um para vários ou zero, onde um *publisher* pode tem ou não vários *contents* associados ao mesmo. A tabela *content* terá uma chave primária denominada *contentId,* que tal como a anterior tabela será um valor único para cada *content* criado.

As tabelas seguintes entram em mais detalhe sobre cada um dos campos.

Tabela 4 – Campos da tabela *subscriber* explicados com mais detalhe.

subscriberId	Chave primária desta tabela, sendo uma coluna do tipo inteiro composta por 11 algarismos	***username***	Utilizado para guardar o username do subscriber. É uma coluna do tipo varchar de tamanho máximo 32 caracteres e é um campo único;
password	Utilizado para guardar a password do subscriber. É uma coluna do tipo varchar de tamanho máximo 32 caracteres;	***firstName***	Utilizado para guardar o 1º nome do subscriber. É uma coluna do tipo varchar de tamanho máximo 32 caracteres;
lastName	Utilizado para guardar o último nome do subscriber. É uma coluna do tipo varchar de tamanho máximo 32 caracteres;	***email***	Utilizado para guardar o email e o nome do subscriber. É uma coluna do tipo varchar de tamanho máximo 750 caracteres e é um campo único;
type	Utilizado para guardar o tipo do subscriber. É uma coluna do tipo inteiro composta por 1 algarismo;	***created_at***	Utilizado para guardar a data e as horas do registo do subscriber. É uma coluna do tipo datetime.

Tabela 5- Campos da tabela *discovery* explicados com mais detalhe.

subscriberId	*Chave estrangeira desta tabela, vinda da tabela adjacente subscriber;*	***discovery_on***	*Utilizado para guardar o se o modo de descoberta do subscriber está on ou não. É uma coluna do tipo inteiro composta por 1 algarismo;*
radius_value	*Utilizado para a distância de descoberta do* subscriber. *É uma coluna do tipo inteiro composta por 2 algarismos;*	***text_on***	*Utilizado para guardar o se o subscriber deseja ou não receber conteúdos de texto. É uma coluna do tipo inteiro composta por 1 algarismo;*
image_on	*Utilizado para guardar o se o* subscriber *deseja ou não receber conteúdos de imagem. É uma coluna do tipo inteiro composta por 1 algarismo;*	***graphics_on***	*Utilizado para guardar o se o subscriber deseja ou não receber conteúdos de gráfico. É uma coluna do tipo inteiro composta por 1 algarismo.*

Tabela 6- Campos da tabela *preferences* explicados com mais detalhe.

subscriberId	Chave estrangeira desta tabela, vinda da tabela adjacente subscriber;	**preferences_on**	Utilizado para guardar o se o modo de preferências do subscriber está on ou não. É uma coluna do tipo inteiro composta por 1 algarismo;
cultural_on	Utilizado para guardar o se o subscriber deseja ou não receber conteúdos do tema cultural. É uma coluna do tipo inteiro composta por 1 algarismo;	**economics_on**	Utilizado para guardar o se o subscriber deseja ou não receber conteúdos do tema economia. É uma coluna do tipo inteiro composta por 1 algarismo;
sports_on	Utilizado para guardar o se o subscriber deseja ou não receber conteúdos do tema desporto. É uma coluna do tipo inteiro composta por 1 algarismo;	**education_on**	Utilizado para guardar o se o subscriber deseja ou não receber conteúdos do tema educação. É uma coluna do tipo inteiro composta por 1 algarismo;
science_on	Utilizado para guardar o se o subscriber deseja ou não receber conteúdos do tema ciência. É uma coluna do tipo inteiro composta por 1 algarismo.		

Tabela 7- Campos da tabela *publisher* explicados com mais detalhe.

publisherId	Chave primária desta tabela, sendo uma coluna do tipo inteiro composta por 11 algarismos;	**username**	Utilizado para guardar o username do publisher. É uma coluna do tipo varchar de tamanho máximo 32 caracteres e é um campo único;
password	Utilizado para guardar a password do publisher. É uma coluna do tipo varchar de tamanho máximo 32 caracteres;	**firstName**	Utilizado para guardar a 1º nome do publisher. É uma coluna do tipo varchar de tamanho máximo 32 caracteres;
lastName	Utilizado para guardar a último nome do publisher. É uma coluna do tipo varchar de tamanho máximo 32 caracteres e é um campo único;	**email**	Utilizado para guardar o email do publisher. É uma coluna do tipo varchar de tamanho máximo 750 caracteres e é um campo único;
email_code	Utilizado para guardar o código para activação do registo do publisher. É uma coluna do tipo varchar de tamanho máximo 32 caracteres;	**active**	Utilizado para guardar se o publisher está activo. É uma coluna do tipo inteiro composta por 1 algarismo;
password_recover	Utilizado para verificar se o pedido recuperação da password do publisher está activa. É uma coluna do tipo inteiro composta por 1 algarismo;	**type**	Utilizado para guardar o tipo do publisher. É uma coluna do tipo inteiro composta por 1 algarismo;
allow_email	Utilizado para guardar se publisher se o publisher aceita receber emails. É uma coluna do tipo inteiro composta por 1 algarismo;	**profile**	Utilizado para guardar o caminho da imagem do publisher. É uma coluna do tipo varchar de tamanho máximo 1024 caracteres.

Tabela 8- Campos da tabela *content* explicados com mais detalhe.

publisherId	*Chave estrangeira desta tabela, vinda da tabela adjacente publisher;*	***contentId***	*Chave primária desta tabela, sendo uma coluna do tipo inteiro composta por 11 algarismos;*
name	*Utilizado para guardar o nome do conteúdo. É uma coluna do tipo* varchar *de tamanho máximo 600 caracteres e é um campo único;*	***path***	*Utilizado para guardar a directoria ou URL do conteúdo. É uma coluna do mediumtext;*
active	*Utilizado para guardar se o conteúdo está activo. É uma coluna do tipo inteiro composta por 1 algarismo;*	***image***	*Utilizado para guardar a directoria ou URL da imagem do conteúdo. É uma coluna do mediumtext;*
type	*Utilizado para guardar o tipo do conteúdo. É uma coluna do tipo* varchar *de tamanho máximo 25 caracteres;*	***theme***	*Utilizado para guardar o tema do conteúdo. É uma coluna do tipo varchar de tamanho máximo 25 caracteres;*
JSONfile	*Utilizado para guardar a directoria,* URL *do conteúdo para ser consumido pelo* subscriber. *É uma coluna do tipo* longtext;	***latitude***	*Utilizado para guardar a latitude do conteúdo. É uma coluna do tipo varchar de tamanho máximo 25 caracteres;*
longitude	*Utilizado para guardar a longitude do conteúdo. É uma coluna do tipo varchar de tamanho máximo 25 caracteres;*	***review***	*Utilizado para guardar mais informações sobre o conteúdo. É uma coluna do tipo texto;*
created_at	*Utilizado para guardar a data e as horas do registo do conteúdo. É uma coluna do tipo* datetime.		

O armazenamento dos dados nas tabelas anteriores permite o correcto funcionamento da *API* e dos dois actores que influenciam o seu preenchimento, o *subscriber* e o *publisher*.

3.4. Estrutura da aplicação móvel (*Android*)

Quando se cria um projecto *Android*, cria-se uma estrutura diferente de pastas do que se está habituado com *Java*. O implementador, alocará todo o código fonte da aplicação na pasta *src*. Por outro lado, outras pastas como a *gen* e a *bin*, são geradas com o propósito do implementador não mexer, pois contém código gerado automaticamente a partir de outras pastas onde o implementador cria e gera os vários recursos da aplicação. As várias dependências do *Android* e bibliotecas referenciadas com seus caminhos encontram-se nas seguintes pastas do projecto:

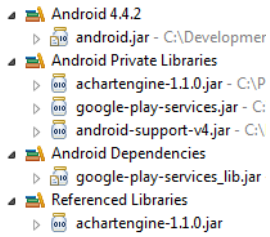


Figura 3-4 Estrutura das dependências e bibliotecas do projecto.

A partir da Figura 3-4 pode-se verificar a utilização das bibliotecas do *Google-play-services*, do *android-support-v4* e *achartengine-1.1.0* já descritas no subcapítulo 2.6.

Neste subcapítulo, abordar-se-á com mais detalhe a estrutura das pastas de implementação desde projecto, nomeadamente o *src* e a *res*, sendo esta a pasta dos recursos da personalização da aplicação. A Figura 3-5 ilustra a divisão do *src* em diferentes pacotes da solução implementada, conforme as suas funções.

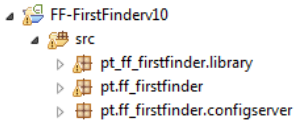


Figura 3-5 Estrutura e divisão por pacotes da pasta *src* da aplicação móvel.

Cada um dos pacotes agregará várias classes ou actividades consoante a sua funcionalidade. A Figura 3-6 ilustra a divisão do *res* em diferentes pastas.

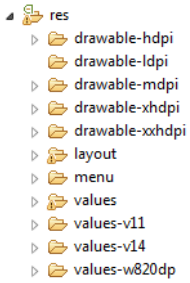


Figura 3-6 Estrutura e divisão por pastas da pasta *res* da aplicação móvel.

3.4.1. src

O *pt_ff_first_finder.library* é um pacote destinado para guardar classes que suportam as diversas actividades da aplicação. Este pacote está dividido da seguinte maneira:

Tabela 9– Detalhes sobre as classes que compõem o pacote *pt_ff_first_finder.library*.

Content.java	É destinada para criar e devolver os vários campos do objecto content para aparecer na lista de pesquisas do utilizador;	***ContentArrayAdapter.java***	É uma lista de objectos content. É utilizada para auxiliar a visualização dos items da lista de pesquisas do utilizador;
DatabaseHandler.java	É utilizada para criar e gerir a tabela que guardará as informações sobre o subscriber no telemóvel;	**DatabaseHandlerContent.java**	É utilizada para criar e gerir a tabela que guardará as informações sobre do content no telemóvel;
GPSTracker.java	É uma implementação do LocationListener. É utilizada para ir buscar a posição do utilizador e ligando e desligando o GPS do utilizador quando necessário;	**JSONParser.java**	É utilizada para tratar do envio e recepção do objecto JSON entre a aplicação e a API;
ReaderCVSfile.java	É utilizada para a recepção, leitura e verificação de erros dos ficheiros to tipo chart;	**UserFunctions.java**	É utilizada para preparar as várias tags e pedidos para a construção do vector JSON.

O *pt.ff_firstfinder* é um pacote destinado a guardar actividades da aplicação. Este pacote está dividido da seguinte maneira:

Tabela 10– Detalhes sobre as classes que compõem o pacote *pt_ff_first_finder.*

ChangePassword.java	Utilizada para lidar com a mudança de password;	***ConsumeContent.java***	Utilizada para lidar com o consumo do conteúdo;
ContactUs.java	Utilizada para lidar com o contacto do suporte da aplicação;	***ContentSettings.java***	Utilizada para lidar com as preferências da pesquisa do conteúdo, nomeadamente o tema;
DiscoveryPreferences.java	Utilizada para lidar com as preferências da descoberta do conteúdo, nomeadamente o tipo e o raio de pesquisa;	***EditProfile.java***	Utilizada para lidar com a modificação de alguns dados do utilizador;
ForgotPassword.java	Utilizada para lidar com o esquecimento do password por parte do utilizador;	***ForgotUsername.java***	Utilizada para lidar com o esquecimento do username por parte do utilizador;
InicialPage.java	Utilizada para lidar com página inicial, onde o utilizador faz a sua autenticação;	***MainApp.java***	Utilizada para lidar com actividade main da aplicação após a autenticação, aqui o utilizador por escolher as diversas formas de pesquisa e edição do seu perfil;
MapFragment.java	Implementação de uma actividade fragmento de OnMapClickListener. Esta actividade está encarregada da criação e gestão do mapa da localização do utilizador e dos seus conteúdos pesquisados;	***NormalSearch.java***	Utilizada para lidar com o display dos conteúdos pesquisados normalmente(a partir do raio escolhido);
OurProfile.java	Utilizada para lidar com a visualização do perfil do utilizador;	***QuickSearch.java***	Utilizada para lidar com a pesquisa e display de conteúdos pelo seu nome;
Registed.java	Utilizada para fazer o display dos dados do utilizador após o registo;	***Register.java***	Utilizada para lidar com o registo do utilizador.

O *pt.ff_firstfinder.configServer* é um pacote destinado para guardar as classes para a configuração do servidor da aplicação. Este pacote está dividido da seguinte maneira:

- *ConfigServer.java:* esta classe é utilizada para guardar os diferentes *URLs* dos pedidos para o servidor, bem como o *URL* do servidor.

3.4.2. res

As várias pastas *drawable* estão destinadas para alocarem as várias imagens utilizadas para personalizar a aplicação. Para a sua visualização das imagens usadas pela aplicação móvel entram-se nos anexos na Tabela 33.

A pasta *layout* guarda os diferentes *layouts* para o ecrã e está dividida da seguinte maneira:

Tabela 11– Detalhes sobre os diferentes *layouts*.

activity_inicial_page.xml	É o primeiro display quando abrimos a aplicação;	changepassword.xml	Aparece quando o utilizador selecciona a opção change password do menu;	contactus.xml	Aparece quando o utilizador selecciona a opção contact us do menu;
content_view.xml	É criado quando o content é consumido dando mais detalhes sobre o mesmo;	contents_list_item	É utilizado para colocar os items na lista depois da pesquisa;	contentsettings.xml	Aparece quando o utilizador selecciona a opção content settings do menu;
discoverypreferences.xml	Aparece quando o utilizador selecciona a opção discovery preferences do menu;	editprofile.xml	É mostrado quando o utilizador escolhe editar o seu perfil;	getuser.xml	É utilizado quando o utilizador esquece-se do seu username;
main.xml	É o display da principal actividade da aplicação, é o display logo após a correcta autenticação do utilizador;	mapdisplay.xml	É utilizado para armazenar o fragmento do mapa;	normalsearch.xml	Aparece depois do utilizador receber os conteúdos da pesquisa normal, ou seja, os conteúdos dentro do raio escolhido;
ourprofile.xml	Aparece quando o utilizador selecciona a opção our profile do menu;	passwordreset.xml	É utilizado quando o utilizador esquece-se do sua password;	quicksearch.xml	Aparece depois do utilizador escolhe fazer uma pesquisa pelo nome;
registed.xml	Aparece depois do utilizador se registar na base de dados;	register.xml	É utilizado para o registo do um novo utilizador na base de dados.		

A pasta *menu* é destinada para guardar os diferentes menus do projecto, e contém a seguinte estrutura:

- *inicial_page.xml:* contém o display do *items* do *menu* da actividade principal da aplicação.

A pasta *values* é destinada a guardar os valores das dimensões, *strings* e estilos utilizados na aplicação.

3.4.3. Tabelas internas criadas para auxiliar a aplicação

Para auxiliar no funcionamento da aplicação, foram criadas duas tabelas internas utilizando a funcionalidade *Android SQLite*. Uma delas encarregar-se-á de guardar as informações do *subscriber*, enquanto a outra guardará os vários conteúdos recebidos do servidor. As suas estruturas estão exibidas no seguinte diagrama *DER* da Figura 3-7.

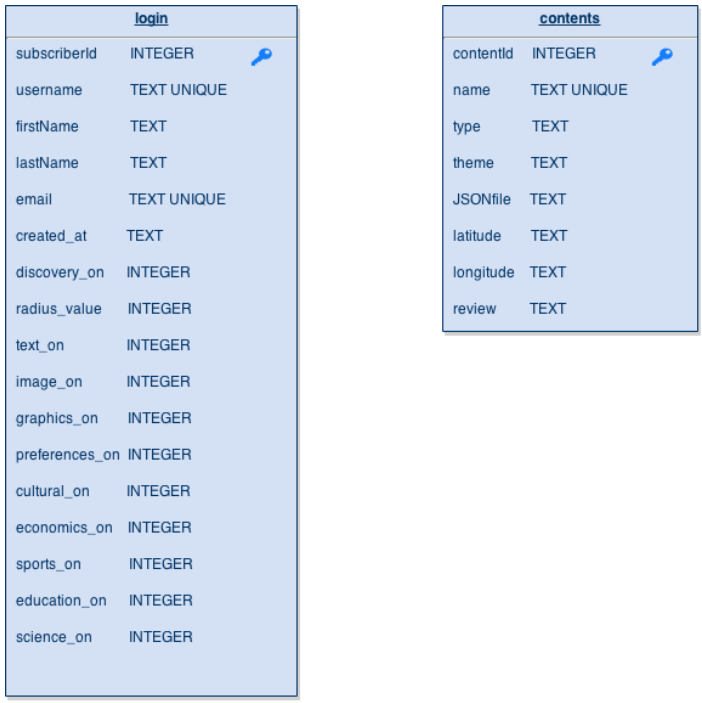


Figura 3-7 Diagrama *DER*, da estrutura das tabelas da base de dados *Android SQLite*.

Tabela 12- Campos da tabela *login* explicados com mais detalhe.

subscriberId	Chave primária desta tabela que guarda o Id do subscriber, sendo uma coluna do tipo inteiro;	***username***	*Utilizada para guardar o* username *do* subscriber, *sendo uma coluna única do tipo* text;
firstName	*Utilizada para guardar o primeiro nome do* subscriber, *sendo uma coluna do tipo* text;	***lastName***	*Utilizada para guardar o último nome do* subscriber, *sendo uma coluna do tipo* text;
email	*Utilizada para guardar o* email *do* subscriber, *sendo uma coluna única do tipo* text;	***created_at***	*Utilizada para guardar o tempo e data da criação do* subscriber, *sendo uma coluna do tipo* text;
discovery_on	*Utilizada para guardar o se o modo de descoberta do* subscriber *está ligado ou não, sendo uma coluna do tipo inteiro;*	***radius_value***	*Utilizada para guardar o valor do raio da descoberta do* subscriber, *sendo uma coluna do tipo inteiro;*
text_on	*Utilizada para guardar o se o* subscriber *deseja receber conteúdos de texto, sendo uma coluna do tipo inteiro;*	***image_on***	*Utilizada para guardar o se o* subscriber *deseja receber conteúdos de imagem, sendo uma coluna do tipo inteiro;*
graphics_on	*Utilizada para guardar o se o* subscriber *deseja receber conteúdos de gráficos, sendo uma coluna do tipo inteiro;*	***cultural_on***	*Utilizada para guardar o se o* subscriber *deseja receber conteúdos do tema cultural, sendo uma coluna do tipo inteiro;*
economics_on	*Utilizada para guardar o se o* subscriber *deseja receber conteúdos do tema económico, sendo uma coluna do tipo inteiro;*	***sports_on***	*Utilizada para guardar o se o* subscriber *deseja receber conteúdos do tema desportivo, sendo uma coluna do tipo inteiro;*
preferences_on	*Utilizada para guardar o se o modo de preferências do* subscriber *está ligado ou não, sendo uma coluna do tipo inteiro;*	***science_on***	*Utilizada para guardar o se o* subscriber *deseja receber conteúdos do tema científico, sendo uma coluna do tipo inteiro;*
education_on	*Utilizada para guardar o se o* subscriber *deseja receber conteúdos do tema educacional, sendo uma coluna do tipo inteiro.*		

Tabela 13- Campos da tabela *content* explicados com mais detalhe.

contentId	*Chave primária desta tabela que guarda o Id do conteúdo, sendo uma coluna do tipo inteiro;*	***name***	*Utilizada para guardar o nome do conteúdo, sendo uma coluna única do tipo* text;
type	*Utilizada para guardar o tipo do conteúdo, sendo uma coluna do tipo* text;	***theme***	*Utilizada para guardar o tema do conteúdo, sendo uma coluna do tipo* text;
JSONfile	*Utilizada para guardar o* URL *do conteúdo, sendo uma coluna do tipo* text;	***latitude***	*Utilizada para guardar a latitude do conteúdo, sendo uma coluna do tipo* text;
longitude	*Utilizada para guardar a longitude do conteúdo, sendo uma coluna do tipo* text;	***review***	*Utilizada para guardar a review do conteúdo, sendo uma coluna do tipo* text.

3.5. Modelo de funcionamento do sistema

Para cada uma das entidades existem protocolos de comunicação diferentes, como referido no subcapítulo 3.1. Neste subcapítulo aprofundar-se-á o conhecimento desses protocolos com recurso a fluxogramas e tabelas para ajudar no entendimento e documentação da *API*.

Inicialmente, informar-se-á sobre o funcionamento do ciclo *publisher,* abordando os principais *HTTP Requests,* as várias verificações feitas pela *API* e pela aplicação *Web-based* e a colocação e edição de conteúdos na base de dados. Em seguida, informar-se-á sobre o funcionamento do ciclo *subscriber,* abordando os diferentes objectos e *arrays* da comunicação *JSON* entre a aplicação móvel e a *API,* as várias verificações feitas tanto na *API* como na aplicação móvel e o pesquisa e consumo de conteúdos da base de dados.

3.5.1. Ciclo *publisher*

3.5.1.1 Registo

O início do ciclo é feito quando o *publisher* deseja registar-se na base de dados. Para melhor entender o seu registo a Figura 3-8 ilustra o fluxograma com as trocas de pedidos e verificações e a Tabela 14 os seus detalhes.

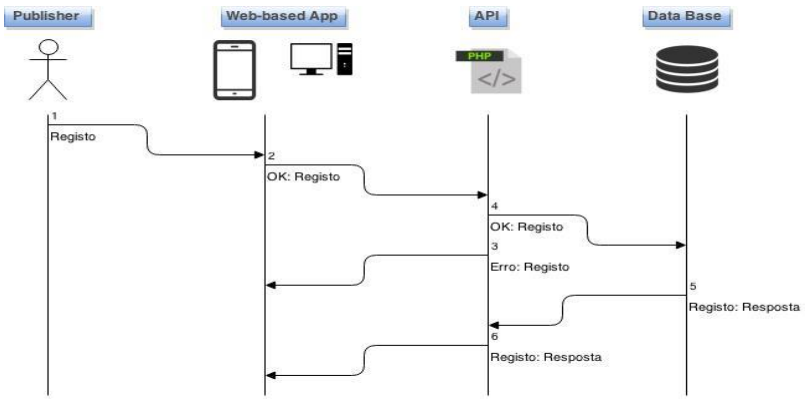


Figura 3-8 Ciclo de mensagens / pedidos entre as entidades e o actor no registo do *publisher*.

Tabela 14 – Descrição mais detalhada das mensagens trocadas entre as entidades e o actor no registo de um *publisher*.

Nº da mensagem	*Descrição*
1	Acção de pedido de registo.
1->2	Verificação se todos os campos obrigatórios estão preenchidos. Caso não estejam, o utilizador terá de preenchê-los e fazer um novo pedido.
2	**1->2 OK**: Pedido de registo, com os dados do *publisher* A. ➔ *HTTP POST "Register" ['username', 'password', 'password_re_type', 'firstName','lastName','email']*
2->3/4	Verificação dos dados de *publisher* A: ▪ *'username'* sem espaços; ▪ 6 < *'password'* < 32 caracteres; ▪ *'password'* = *'password_re_type'*; ▪ *'email'* válido (*online*).
3	**2->3/4 ERRO:** aviso de erros nos dados, imprimindo-os no ecrã do utilizador.
4	**2->3/4 OK:** pedido de *update* do *publisher* na base de dados.
5	Resposta ao pedido: ➔ **OK:** Sucesso no registo. ➔ **Erro:** *'email'* ou *'username'* já existentes na base de dados.
6	Resposta ao pedido: ➔ **OK:** Sucesso no registo. ➔ **Erro:** impressão no ecrã do erro.

3.5.1.2 Recuperar *password* ou *username*

O *publisher* pode se esquecer dos seus dados de autenticação, então para esses casos existe esta opção de escolha. Para melhor entender esta opção a Figura 3-9 ilustra o fluxograma com as trocas de pedidos e verificações e a Tabela 15 os seus detalhes.

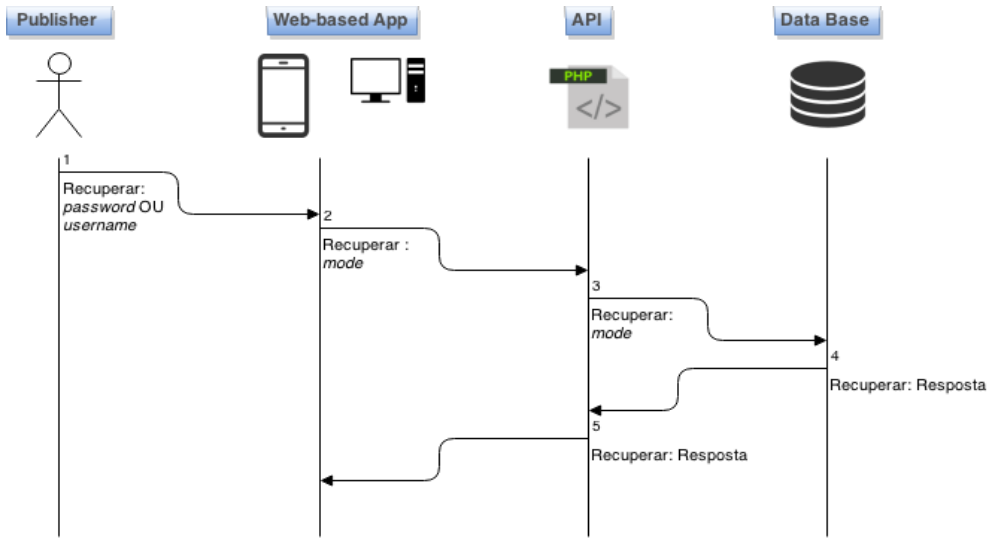


Figura 3-9 Ciclo de mensagens / pedidos entre as entidades e o actor na recuperação da *password* ou do *username* do *publisher*.

Tabela 15 – Descrição mais detalhada das mensagens trocadas entre as entidades e o actor na recuperação da *password* ou do *username* do *publisher*.

Nº da mensagem	Descrição
1	Acção de pedido de recuperação *username* ou *password*.
1->2	Verificação do *mode* de recuperação escolhido. ➔ *HTTP GET "mode" = 'username' ou 'password'*
2	Pedido de recuperação do "*mode*", com a passagem do campo '*email*' *publisher* A. ➔ *HTTP POST "email" ['email']*
2->3	Verificação que o campo '*email*' não está vazio.
3	Pedido de verificação se o *email* existe na base de dados.
4	Resposta ao pedido: ➔ **OK:** Existe. ➔ **Erro:** '*email*' não existente na base de dados.
5	Resposta ao pedido: ➔ **OK:** Envio dos dados para o *email*. Sucesso na recuperação. ➔ **Erro:** impressão no ecrã do erro.

3.5.1.3 Login

O *publisher* para poder usufruir de todo o potencial da aplicação *Web-based* deverá fazer a autenticação da sua conta, *login*, após o seu registo. Para melhor entender esta opção a Figura 3-10 ilustra o fluxograma com as trocas de pedidos e verificações e a Tabela 16 os seus detalhes.

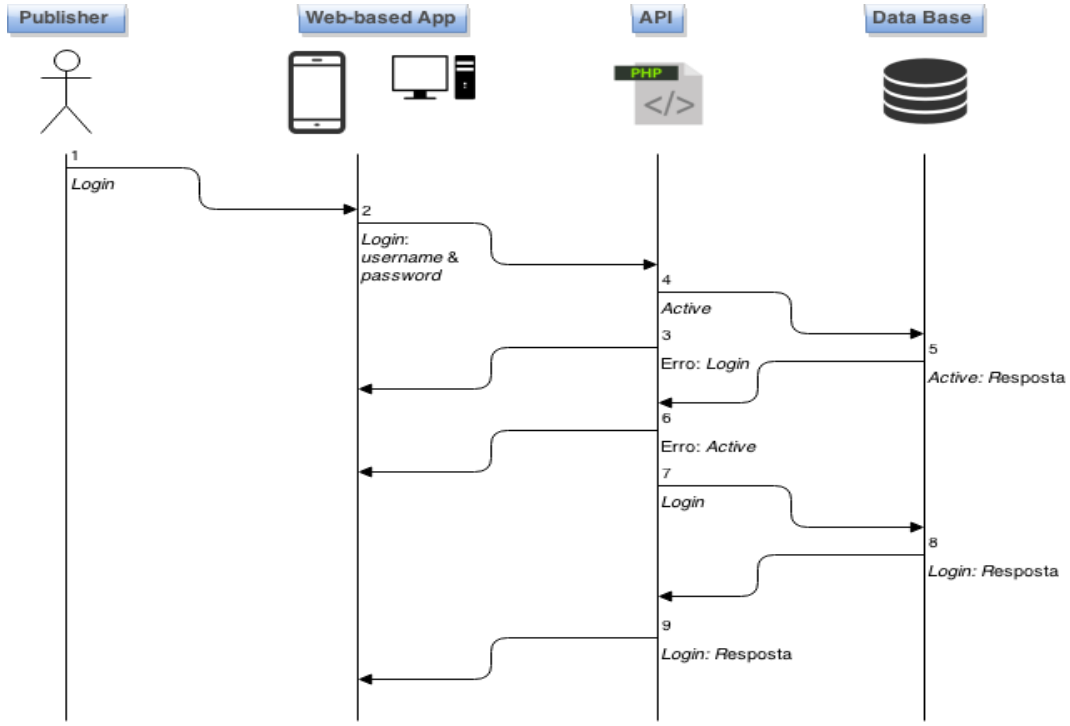


Figura 3-10 Ciclo de mensagens / pedidos entre as entidades e o actor no *login* do *publisher*.

Tabela 16 – Descrição mais detalhada das mensagens trocadas entre as entidades e o actor no *login* do *publisher*.

Nº da mensagem	Descrição
1	Acção de pedido de *login* do *publisher*
2	Pedido de login com os dados '*username*' e '*password*' do *publisher*. ➔ HTTP POST " Log in" ['*username*', '*password*']
2->3/4	Verificação de dados "Log in": ▪ '*username*' e '*password*' vazios; ▪ '*password*' > 32 caracteres.
3	**2->3/4 ERRO:** aviso de erros nos dados.
4	**2->3/4 OK:** pedido de confirmação de activação do '*username*'.
5	Resposta ao pedido: ➔ **OK:** Sucesso no registo. ➔ **Erro:** '*username*' não existe ou não está activo.
6	Aviso de '*username*' não existente ou não activo.
7	Verificar se existe algum *publisher* com este '*username*' e '*password*'.
8	Resposta ao pedido: ➔ **OK:** Existe um *publisher*. ➔ **Erro:** Não existe nenhum *publisher* com esses dados.
9	Resposta ao pedido: ➔ **OK:** Passar a sessão do *publisher* a *login*. ➔ **Erro:** A combinação *username/password* está incorrecta.

Após o *publisher* fazer o *login*, uma *array* com os seus dados será mantida até este fazer o *logout*.

3.5.1.4 Profile & Settings

O *publisher*, após autenticado, pode querer ver ou mudar o seu perfil. Para melhor entender a opção de visualização de perfil a Figura 3-11 ilustra o fluxograma com as trocas de pedidos e verificações e a Tabela 17 os seus detalhes.

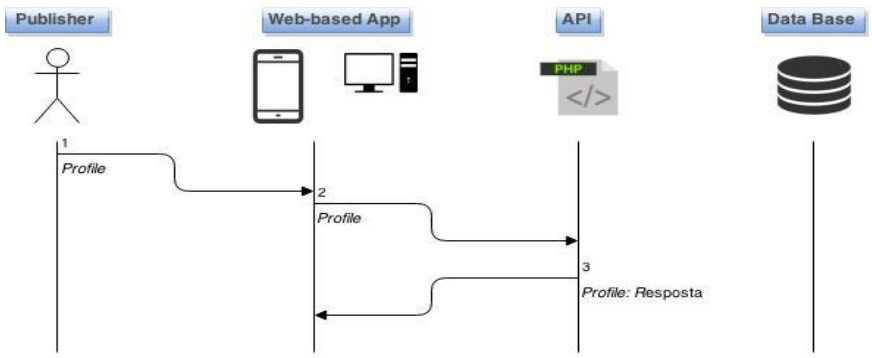


Figura 3-11 Ciclo de mensagens / pedidos entre as entidades e o actor na visualização do perfil do *publisher*.

Tabela 17 - Descrição mais detalhada das mensagens trocadas entre as entidades e o actor na visualização do perfil do *publisher*.

Nº da mensagem	Descrição
1	Acção de pedido de ver o *profile* do *publisher*
2	Pedido de ver o *profile* do *publisher* com o *'username'*. ➔ *HTTP GET "…/FF_first_finderV2/'username'"*
3	Devolução de alguns dos dados da *array* do *publisher* guardada no login está *login*.

Para melhor entender a opção de modificação de perfil a Figura 3-12 ilustra o fluxograma com as trocas de pedidos e verificações e a Tabela 18 os seus detalhes.

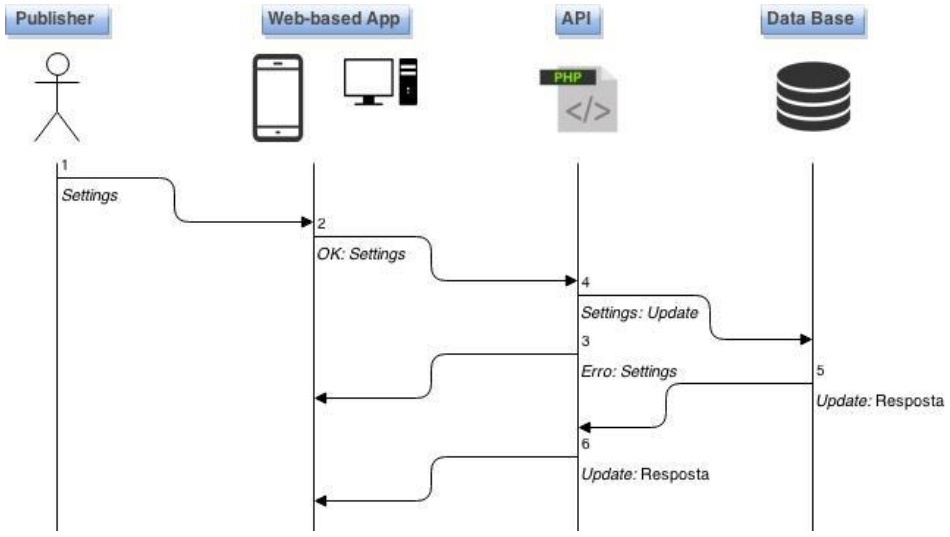


Figura 3-12 Ciclo de mensagens / pedidos entre as entidades e o actor na modificação do perfil do *publisher*.

Tabela 18- Descrição mais detalhada das mensagens trocadas entre as entidades e o actor na modificação do perfil do *publisher*.

Nº da mensagem	Descrição
1	Acção de pedido de modificar o *profile* do *publisher*
2	Pedido de modificar o *profile* do *publisher* com os dados a actualizar. ➔ *HTTP POST "Update" ['firstName','lastName','email','allow_email']*
2->3/4	Verificar dados: ▪ Todos campos obrigatórios estão preenchidos; ▪ 'email' válido (online).
3	**2->3/4 ERRO:** aviso de erros nos dados.
4	**2->3/4 OK:** pedido de *update* do *publisher* na base de dados.
5	Resposta ao pedido: ➔ **OK:** *Update* com sucesso. ➔ **Erro:** *'email'* já existe na base de dados.
6	Resposta ao pedido: ➔ **OK:** *Update* com sucesso. ➔ **Erro:** *'email'* já existe na base de dados.

3.5.1.5 Mudar a *password*

O *publisher*, após autenticado, pode querer mudar a sua *password*. Para melhor entender a opção de modificação de *password* a Figura 3-13 ilustra o fluxograma com as trocas de pedidos e verificações e a Tabela 19 os seus detalhes.

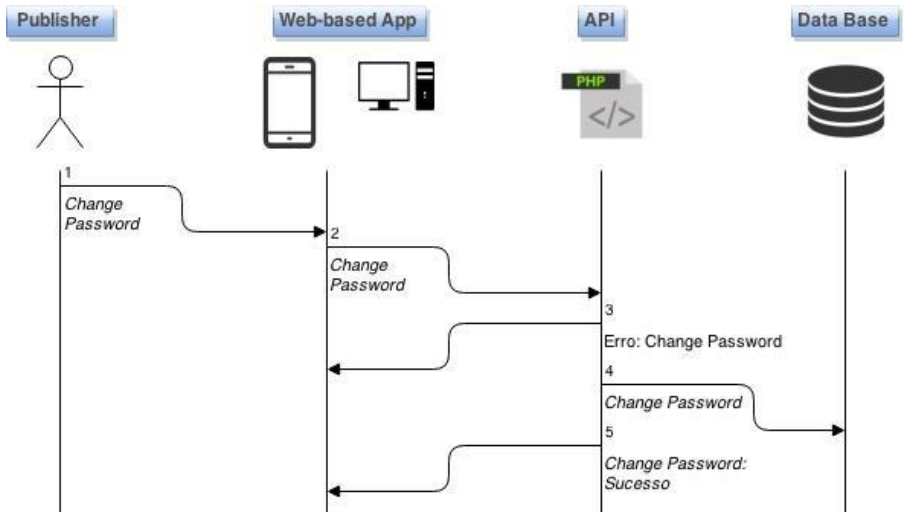


Figura 3-13 Ciclo de mensagens / pedidos entre as entidades e o actor na modificação da *password* do publisher.

Tabela 19 - Descrição mais detalhada das mensagens trocadas entre as entidades e o actor na modificação da *password* do *publisher*.

Nº da mensagem	*Descrição*
1	Acção de pedido de mudança de *password*.
2	Pedido de mudança de *'password'*, com a passagem da nova *'password' publisher* A. ➔ *HTTP POST "Change password" ['current_password','password', 'password_again']*
2->3/4	Verificação dos dados: ▪ *' current_password'* = *password* do *publisher*; ▪ 6 < *'password'* < 32 caracteres; ▪ *'password'* = *'password_again'*.
3	**2->3/4 ERRO:** aviso de erros nos dados.
4	**2->3/4 OK:** pedido de *update* da *password* do *publisher* na base de dados.
5	Sucesso na mudança de *password*.

3.5.1.6 Mudar a foto de perfil

O *publisher*, após autenticado, pode querer mudar a sua foto de perfil. Para melhor entender a opção de modificação de foto a Figura 3-14 ilustra o fluxograma com as trocas de pedidos e verificações e a Tabela 20 os seus detalhes.

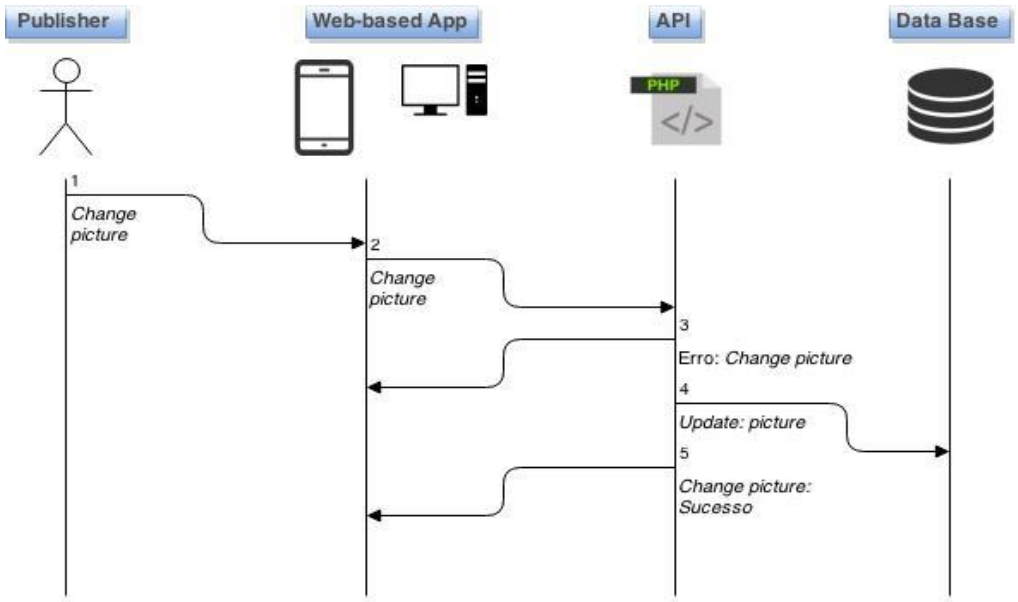


Figura 3-14 Ciclo de mensagens / pedidos entre as entidades e o actor na modificação da foto de perfil do publisher.

Tabela 20 - Descrição mais detalhada das mensagens trocadas entre as entidades e o actor na modificação da foto de perfil do publisher.

Nº da mensagem	*Descrição*
1	Acção de pedido de mudança de foto do perfil.
2	Pedido de mudança de foto do perfil, com a passagem da nova foto *publisher*. ➔ *HTTP POST "profile" ['ficheiro']*
2->3/4	Verificação dos dados: ▪ Se a extensão do ficheiro é permitida.
3	**2->3/4 ERRO:** aviso de erros no ficheiro escolhido.
4	**2->3/4 OK:** pedido de *update* do campo *'profile'* do *publisher* na base de dados.
5	Sucesso na mudança de foto do perfil.

3.5.1.7 Adicionar conteúdos

O *publisher*, após autenticado, pode querer adicionar conteúdos. Para melhor entender a opção de adição de conteúdos a Figura 3-15 ilustra o fluxograma com as trocas de pedidos e verificações e a Tabela 21 os seus detalhes.

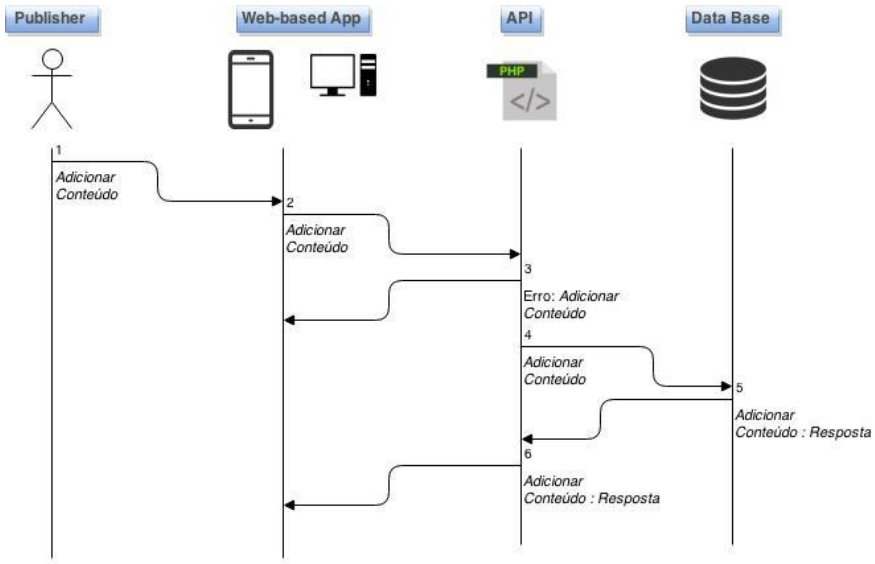


Figura 3-15 Ciclo de mensagens / pedidos entre as entidades e o actor na adição de um conteúdo de um *publisher*.

Tabela 21 - Descrição mais detalhada das mensagens trocadas entre as entidades e o actor na adição de um conteúdo de um *publisher*.

Nº da mensagem	*Descrição*
1	Acção de pedido para adicionar um conteúdo.
2	Pedido para adicionar um conteúdo, com a passagem de parâmetros: ➔ *HTTP POST "Add content" [' name',' file_path',' URL_path',' latitude',' longitude',' theme',' review']*
2->3/4	Verificação dos dados do conteúdo: ▪ Se todos os campos obrigatórios estão preenchidos; ▪ Se o URL existe (caso o conteúdo seja um URL); ▪ Verificar o tamanho máximo do ficheiro (1MB); ▪ Verificar se o ficheiro é permitido tem a extensão permitida para a sua publicação; ▪ Verificar se o nome do conteúdo já existe; ▪ Verificar se o ficheiro já existe (caso o conteúdo seja um ficheiro para guardar no servidor); ▪ Verificar se o ficheiro for do tipo *chart*, se se encontra dentro das especificações necessárias.
3	**2->3/4 ERRO:** aviso de erros.
4	**2->3/4 OK:** pedido para adicionar um conteúdo do *publisher* na base de dados.
5	Resposta ao pedido: ➔ **OK:** Sucesso a adicionar do conteúdo; ➔ **Erro:** conteúdo com o nome já existente.
6	Resposta ao pedido: ➔ **OK:** Sucesso a adicionar do conteúdo; ➔ **Erro:** conteúdo com o nome já existente.

3.5.1.8 Visualização, edição e eliminação de conteúdos

O *publisher*, após autenticado, pode ver a lista dos seus conteúdos para depois editá-los ou eliminá-los.

Para melhor entender a opção de visualização de conteúdos a Figura 3-16 ilustra o fluxograma com as trocas de pedidos e verificações e a Tabela 22 os seus detalhes.

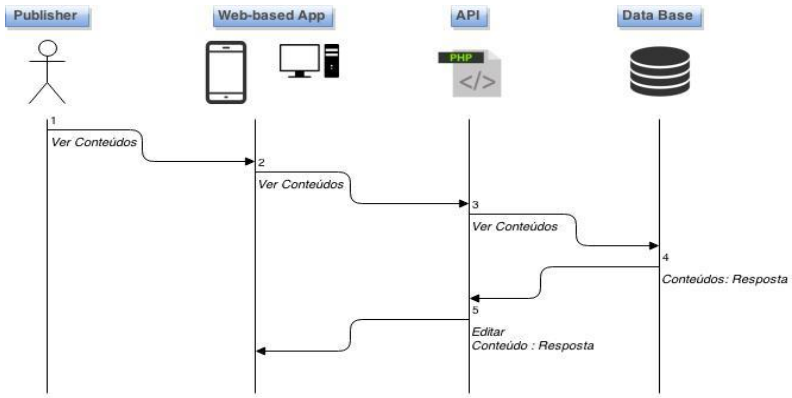


Figura 3-16 Ciclo de mensagens / pedidos entre as entidades e o actor na visualização de conteúdos de um *publisher*.

Tabela 22 - Descrição mais detalhada das mensagens trocadas entre as entidades e o actor na visualização de conteúdos de um *publisher*.

Nº da mensagem	*Descrição*
1	Acção de pedido de visualização dos conteúdos do *publisher*.
2	Pedido de visualização de os conteúdos do *publisher*. Não é necessário um *HTTP request*, porque a *API* já tem todas as informações necessárias sobre o publisher em questão.
3	Ver todos os conteúdos do *publisher* com o *'publisherId'*.
4	Devolve os conteúdos do *publisher*.
5	Devolve os conteúdos do *publisher*.

Após a visualização dos conteúdos, o *publisher* pode escolher entre editar ou eliminar o conteúdo. Para melhor entender a opção de edição de um conteúdo a Figura 3-17 ilustra o fluxograma com as trocas de pedidos e verificações e a Tabela 23 os seus detalhes.

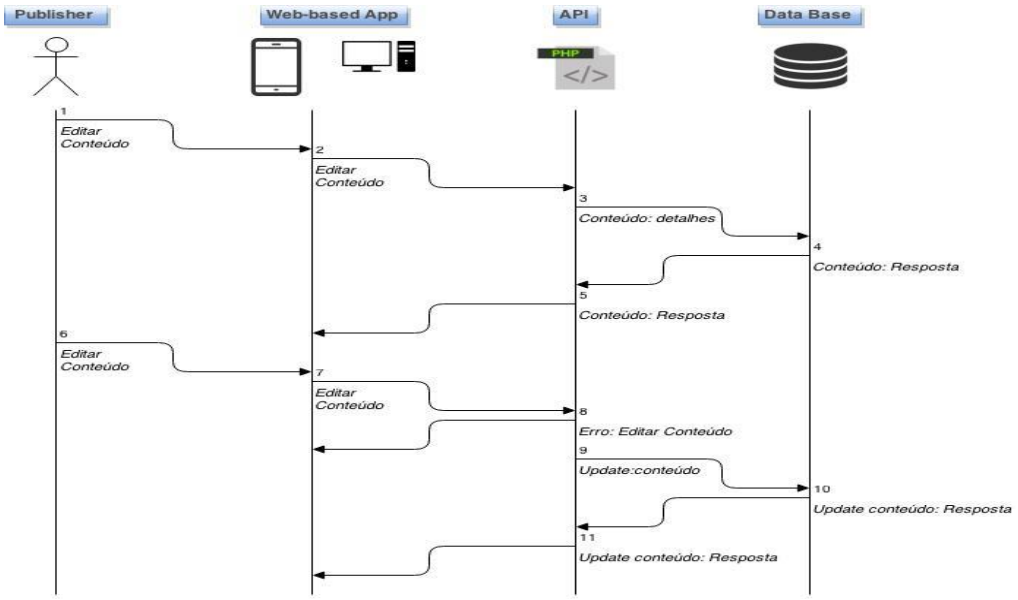


Figura 3-17 Ciclo de mensagens / pedidos entre as entidades e o actor na edição de conteúdos de um *publisher*.

Tabela 23 - Descrição mais detalhada das mensagens trocadas entre as entidades e o actor na edição de conteúdos de um *publisher*.

Nº da mensagem	*Descrição*
1	Acção de pedido para editar um conteúdo.
2	Pedido para editar um conteúdo, com a passagem de parâmetros: ➔ *HTTP GET "…/FF_first_finderV2/edit_content.php?EditButton=Edit+->'nome do conteúdo'"*
3	Pedir detalhes do conteúdo à base de dados.
4	Envio de detalhes do conteúdo.
5	Envio de detalhes do conteúdo.
6	Acção de modificar o conteúdo.
7	Pedido para editar um conteúdo, com a passagem de parâmetros a editar: ➔ *HTTP POST "uploadContent" ['file_path', 'URL_path',' theme', 'active',' review']*
7->8/9	Verificação dos dados do conteúdo: ▪ Se todos os campos obrigatórios estão preenchidos; ▪ Se o URL existe (caso o conteúdo seja um URL); ▪ Verificar o tamanho máximo do ficheiro (1MB); ▪ Verificar se o ficheiro é permitido tem a extensão permitida para a sua publicação; ▪ Verificar se o nome do conteúdo já existe; ▪ Verificar se o ficheiro já existe (caso o conteúdo seja um ficheiro para guardar no servidor); ▪ Verificar se o ficheiro for do tipo *chart*, se se encontra dentro das especificações necessárias.
8	**7->8/9 OK:** aviso de erros.
9	**7->8/9 OK:** pedido de *upload* do conteúdo do *publisher* na base de dados.
10	Resposta ao pedido: ➔ **OK:** Sucesso do *upload* do conteúdo; ➔ **Erro:** Insucesso *update*.
11	Resposta ao pedido: ➔ **OK:** Sucesso do *upload* do conteúdo; ➔ **Erro:** Insucesso *update*.

Para melhor entender a opção de eliminação de um conteúdo a Figura 3-18 ilustra o fluxograma com as trocas de pedidos e verificações e a Tabela 24 os seus detalhes.

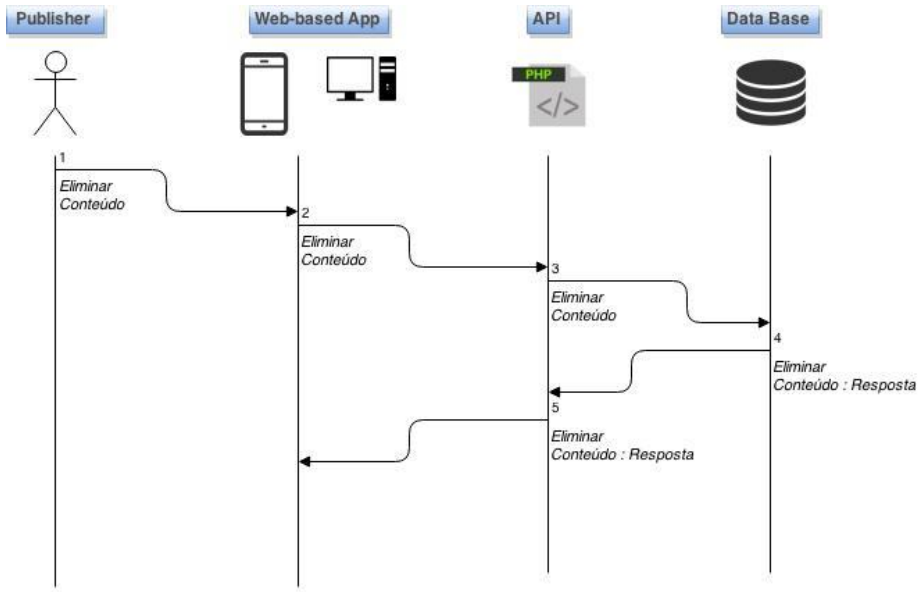


Figura 3-18 Ciclo de mensagens / pedidos entre as entidades e o actor na eliminação de conteúdos de um *publisher*.

Tabela 24 - Descrição mais detalhada das mensagens trocadas entre as entidades e o actor na eliminação de conteúdos de um *publisher*.

Nº da mensagem	*Descrição*
1	Acção de pedido para eliminar um conteúdo.
2	Pedido para eliminar um conteúdo: ➔ *HTTP GET* *"…/FF_first_finderV2/edit_content.php?DeleteButton=Delete+->'nome do conteúdo'"*
3	Apagar o conteúdo com o nome = *'nome do conteúdo'*.
4	Resposta ao pedido: ➔ **OK:** Sucesso de remoção do conteúdo; ➔ **Erro:** Insucesso de remoção.
5	Resposta ao pedido: ➔ **OK:** Sucesso de remoção do conteúdo; ➔ **Erro:** Insucesso de remoção.

3.5.2. Ciclo *Subscriber*

3.5.2.1 Registo

O início do ciclo é feito quando o *subscriber* deseja registar-se na base de dados. Para melhor entender o seu registo a Figura 3-19 ilustra o fluxograma com as trocas de pedidos e verificações e a Tabela 25 os seus detalhes.

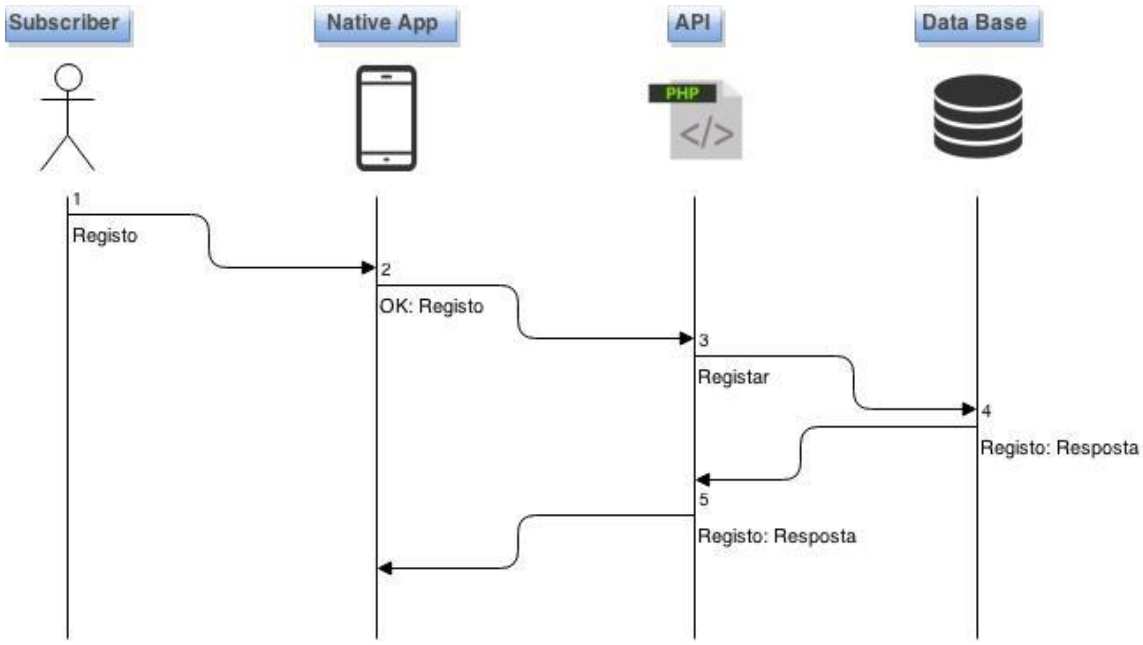


Figura 3-19 Ciclo de mensagens / pedidos entre as entidades e o actor no registo do *subscriber*.

Tabela 25 – Descrição mais detalhada das mensagens trocadas entre as entidades e o actor no registo de um *subscriber*.

Nº da mensagem	*Descrição*
1	Acção de pedido de registo.
1->2	Verificação dos dados do *subscriber*: ▪ Campos obrigatórios todos preenchidos; ▪ *'username'* > 4 caracteres ; ▪ *'password'* = *'password_again'*; ▪ *'password'* > 5 caracteres; ▪ Se está ligado à internet.
2	**1->2 OK**: Pedido de registo, com os dados do *subscriber*. ➔ *JSON { "tag" : "register" , " username" : "nome" , " password" :" value" , "firstName" : "1º nome" , "lastName" : "sobrenome" , "email" : "email" }*
3	Pedido de registo do *subscriber* recebido na tabela *subscriber* e colocação dos campos predefinidos nas tabelas *preferences* e *discovery*.
4	Resposta ao pedido: ➔ **OK:** Sucesso no registo. ➔ **Erro:** Não se conseguiu registar.

5	Resposta ao pedido para o *subscriber*, enviando a resposta na seguinte *String JSON*:
	➔ **OK:** Sucesso no registo.
	Exemplo:
	`{"tag":"register","success":1,"error":0,"subscriber":{"subscriberId":"31","username":"mestre1","firstName":"aloh","lastName":"bora","email":"gonalo.mestre@yahoo.com","created_at":"2015-03-13 20:17:29"}}`
	➔ **Erro:** Não se conseguiu registar.
	`{"tag":"register","success":0,"error":1,"error_msg":"JSON Error occurred in Registration"}`
	`{"tag":"register","success":0,"error":2,"error_msg":"User already existed"}`
	`{"tag":"register","success":0,"error":3,"error_msg":"Invalid Email"}`
	`{"tag":"register","success":0,"error":5,"error_msg":"JSON ERROR"}`

3.5.2.2 Recuperar *password* ou *username*

O *subscriber* pode-se esquecer dos seus dados de autenticação. Então para esses casos existe esta opção de escolha. Para melhor entender esta opção a Figura 3-20 ilustra o fluxograma com as trocas de pedidos e verificações e a Tabela 26 os seus detalhes.

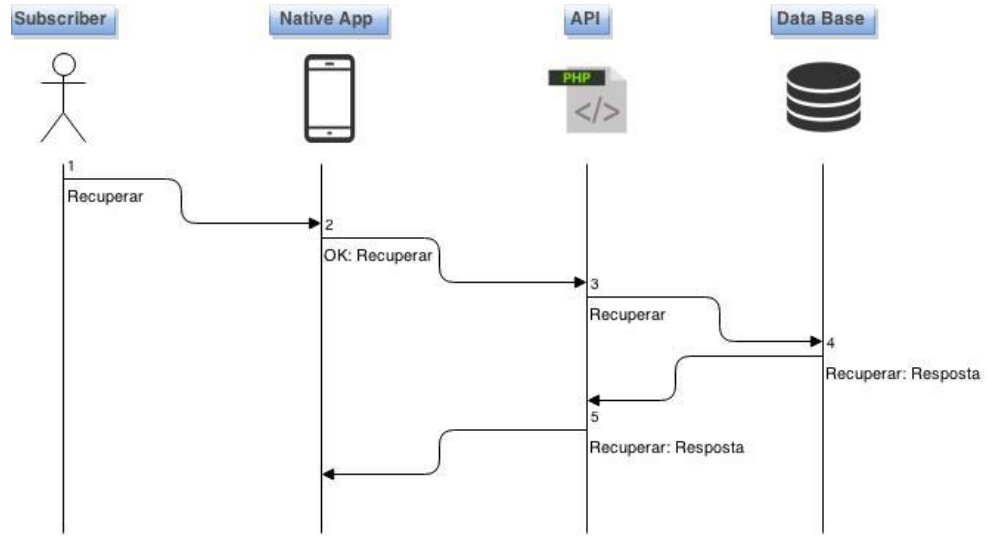


Figura 3-20 Ciclo de mensagens / pedidos entre as entidades e o actor na recuperação da *password* ou do *username* do *subscriber*.

Tabela 26 – Descrição mais detalhada das mensagens trocadas entre as entidades e o actor na recuperação da *password* ou do *username* do *subscriber*.

Nº da mensagem	*Descrição*
1	Acção de pedido de recuperação de *username* ou *password*.
1->2	Verificação dos dados do *subscriber*: ▪ Campos obrigatórios todos preenchidos; ▪ Se está ligado à internet.
2	**1->2 OK**: Pedido de recuperação de *username* ou *password* do *subscriber*. ➔ *JSON { "tag" : "forgotuser", " forgotuser" : "email" }* ➔ *JSON { "tag" : "forgotpass", " forgotpassword" : "username" }*
3	Pedir dados do *subscriber* pelo *email* no caso da recuperação do *username;* Pedir dados do *subscriber* pelo *username* no caso da recuperação da *password;*
4	Resposta ao pedido: ➔ **OK:** Dados do *subscriber*. ➔ **Erro:** Não conseguiu encontrar o *subscriber*.

5	Resposta ao pedido para o *subscriber*, enviando a resposta na seguinte *String JSON*:
	➔ **OK:** Enviar um *email* para o *subscriber* com a informação. No caso da *password* é criada uma *password* nova.
	Exemplo:
	`{"tag":"forgotuser","success":1,"error":0}`
	`{"tag":"forgotpass","success":1,"error":0}`
	➔ **Erro:** Não conseguiu encontrar o *subscriber*.
	`{"tag":"forgotuser","success":0,"error":1,"error_msg":""}`
	`{"tag":"forgotuser","success":0,"error":2,"error_msg":"Email not exist"}`
	`{"tag":"forgotuser","success":0,"error":5,"error_msg":"JSON ERROR"}`
	`{"tag":"forgotpass","success":0,"error":1,"error_msg":""}`
	`{"tag":"forgotpass","success":0,"error":2,"error_msg":"User not exist"}`
	`{"tag":"forgotpass","success":0,"error":5,"error_msg":"JSON ERROR"}`

3.5.2.3 Login

O *subscriber* para poder usufruir de todo o potencial da aplicação *Native App* deverá fazer a autenticação da sua conta, *login*, após o seu registo. Para melhor entender esta opção a Figura 3-21 ilustra o fluxograma com as trocas de pedidos e verificações e a Tabela 27 os seus detalhes.

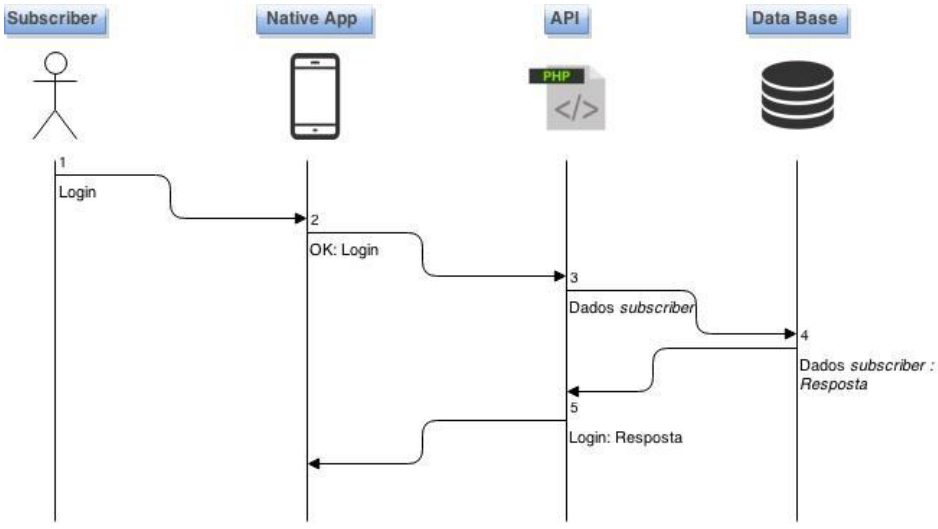


Figura 3-21 Ciclo de mensagens / pedidos entre as entidades e o actor no *login* do *subscriber*.

Tabela 27 - Descrição mais detalhada das mensagens trocadas entre as entidades e o actor no *login* do *publisher*.

Nº da mensagem	*Descrição*
1	Acção de pedido de *login*.
1->2	Verificação dos dados do *subscriber*: ▪ Campos obrigatórios todos preenchidos; ▪ Se está ligado à internet.
2	**1->2 OK**: Pedido de registo, com os dados do *subscriber*. ➔ *JSON { "tag" : "login" , " username" : "nome" , " password" :" value" }*
3	Pedido de *login* do *subscriber*, recebendo o os detalhes do *subscriber username* e *password* iguais à *String JSON*.
4	Resposta ao pedido: ➔ **OK:** *Subscriber* existente, detalhes enviados. ➔ **Erro:** Não se conseguiu registar.
5	Resposta ao pedido para o *subscriber*, enviando a resposta na seguinte *String JSON*: ➔ **OK:** Sucesso no login. Exemplo: `{"tag":"login","success":1,"error":0,"subscriber":{"subscriberId":"28","username":"mestre","firstName":"Goncalo","lastName":"Mestre","email":"goncalo.mes@gmail.com","created_at":"2015-01-26 16:21:36","discovery_on":"1","radius_value":"12","text_on":"1","image_on":"1","graphics_on":"1","preferences_on":"1","cultural_on":"1","economics_on":"1","sports_on":"1","education_on":"1","science_on":"1"}}` ➔ **Erro:** *login* com insucesso. `{"tag":"login","success":0,"error":1,"error_msg":"Incorrect username or password!"}` `{"tag":"login","success":0,"error":5,"error_msg":"JSON ERROR"}`

3.5.2.4 Mudar a *password*

O *subscriber*, após autenticado, pode querer mudar a sua *password*. Para melhor entender a opção de modificação de *password* a Figura 3-22 ilustra o fluxograma com as trocas de pedidos e verificações e a Tabela 28 os seus detalhes.

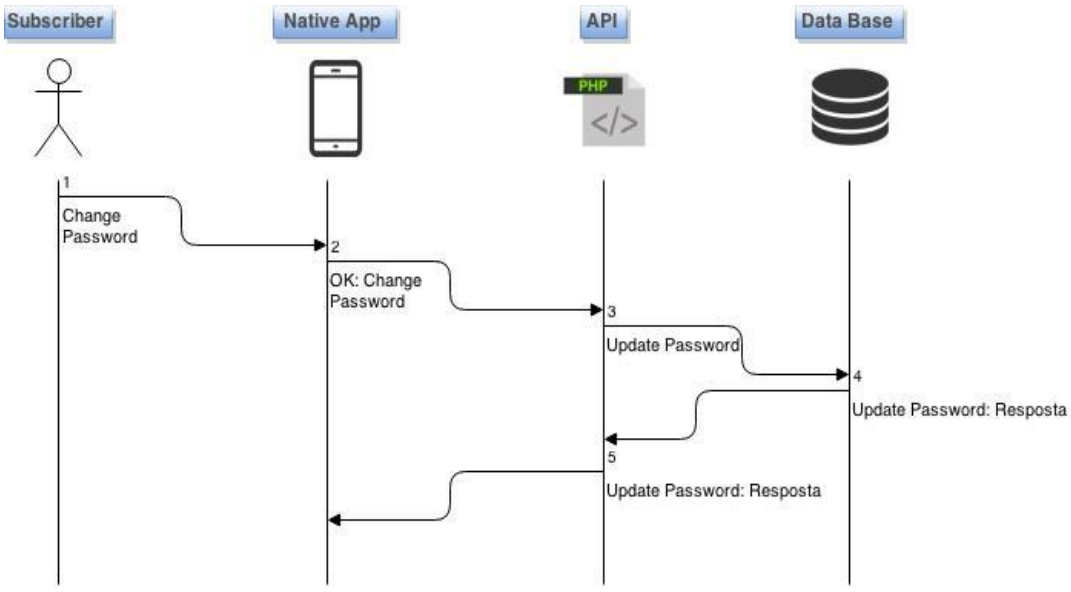


Figura 3-22 Ciclo de mensagens / pedidos entre as entidades e o actor na modificação da *password* do *subscriber*.

Tabela 28 - Descrição mais detalhada das mensagens trocadas entre as entidades e o actor na modificação da *password* do *subscriber*.

Nº da mensagem	*Descrição*
1	Acção de pedido de mudar a *password*.
1->2	Verificação dos dados do *subscriber*: ▪ *password* > 5 caracteres; ▪ Se está ligado à internet.
2	**1->2 OK**: Pedido de registo, com os dados do *subscriber*. ➔ *JSON { "tag" : "changepass" , " username" : "nome" , " newpas" :" value" }*
3	Pedido de mudar a *password* do *subscriber*, recebendo o os detalhes do *subscriber username* e *newpass* iguais à *String JSON*.
4	Resposta ao pedido: ➔ **OK:** Sucesso na modificação da *password*. ➔ **Erro:** Mudança de *password* com insucesso.

5	Resposta ao pedido para o *subscriber*, enviando a resposta na seguinte *String JSON*:
	➔ **OK:** Sucesso na modificação da *password*.
	Exemplo:
	`{"tag":"changepass","success":1,"error":0}`
	➔ **Erro:** Mudança de *password* com insucesso.
	`{"tag":"changepass","success":0,"error":1,"error_msg":""}`
	`{"tag":"changepass","success":0,"error":2,"error_msg":"User not exist"}`
	`{"tag":"changepass","success":0,"error":5,"error_msg":"JSON ERROR"}`

3.5.2.5Logout

O *subscriber*, após autenticado, pode desejar fazer o *logout*. Para melhor entender a opção de *logout* a Figura 3-23 ilustra o fluxograma com as trocas de pedidos e verificações e a Tabela 29 os seus detalhes.

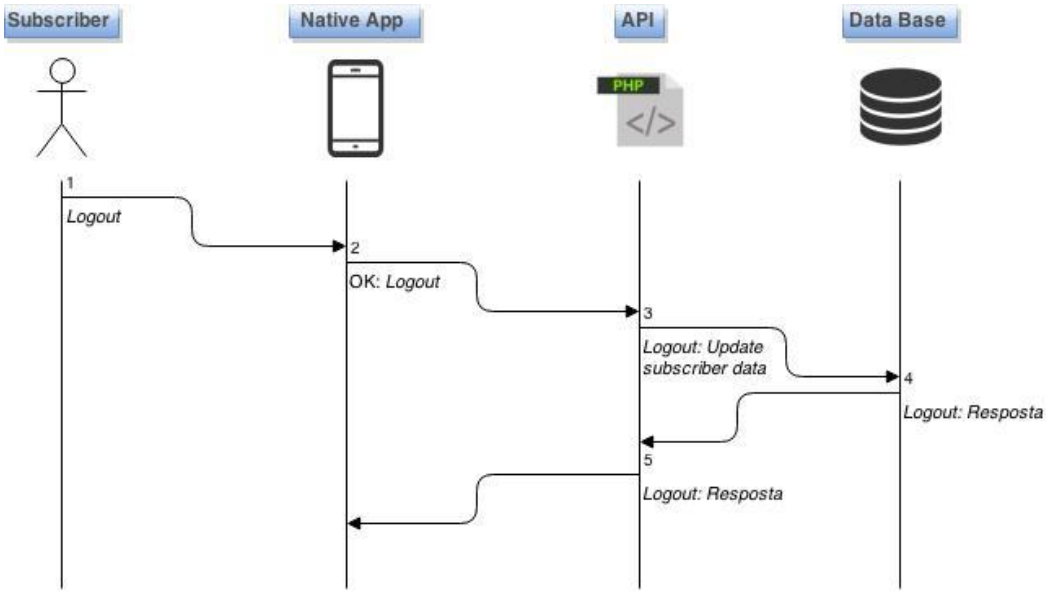


Figura 3-23 Ciclo de mensagens / pedidos entre as entidades e o actor no *logout* do *subscriber*.

Tabela 29 - Descrição mais detalhada das mensagens trocadas entre as entidades e o actor no *logout* do *subscriber*.

Nº da mensagem	Descrição
1	Acção de pedido de *logout*.
1->2	Verificação dos dados do *subscriber*: ▪ Se está ligado à internet. Se não estiver ligado à internet, o *logout* é feito sem guardar os dados.
2	**1->2 OK**: Pedido de registo, com os dados do *subscriber*. ➔ *JSON { "tag" : "logout" , " username" : "nome" , " firstName" :" 1º nome", " lastName" :" sobrenome"," discovery_on" :" activo ou não"," radius_value" :" valor do raio de descoberta"," texto_on" :" :" activo ou não "," image_on" :" :" activo ou não "," graphics_on" :" :" activo ou não "," preferences_on" :":" activo ou não "," cultural_on" :" :" activo ou não "," economics_on" :" activo ou não ", " sports_on" :" activo ou não ", " education_on" :" activo ou não ", " science_on" :" activo ou não " }*
3	Pedido de *update* dos campos da *String JSON* das tabelas *subscriber, discovey* e *preferences*.
4	Resposta ao pedido: ➔ **OK:** Sucesso no *update* das tabelas. ➔ **Erro:** *Update* das tabelas com insucesso.

5	Resposta ao pedido para o *subscriber*, enviando a resposta na seguinte *String JSON*:
	➔ **OK:** Sucesso no *update* das tabelas.
	Exemplo:
	`{"tag":"logout","success":1,"error":0}`
	➔ **Erro:** *Update* das tabelas com insucesso.
	`{"tag":"logout","success":0,"error":1,"error_msg":" JSON Error occurred in Logout "}`
	`{"tag":"logout","success":0,"error":2,"error_msg":"User not exist"}`
	`{"tag":"logout","success":0,"error":5,"error_msg":"JSON ERROR"}`

3.5.2.6Fazer uma pesquisa normal (a partir do raio)

O *subscriber*, após autenticado, pode desejar fazer uma pesquisa dos conteúdos de seu interesse dentro de um raio escolhido pelo mesmo. Para melhor entender essa opção de pesquisa a Figura 3-24 ilustra o fluxograma com as trocas de pedidos e verificações e a Tabela 30 os seus detalhes.

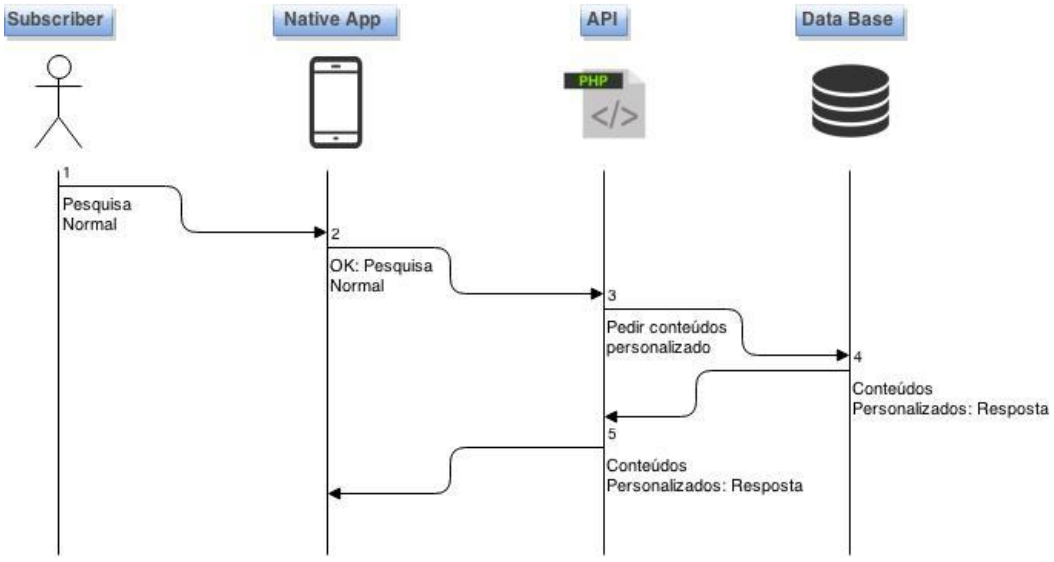


Figura 3-24 Ciclo de mensagens / pedidos entre as entidades e o actor na pesquisa normal do *subscriber*.

Tabela 30 - Descrição mais detalhada das mensagens trocadas entre as entidades e o actor na pesquisa normal do *subscriber*.

Nº da mensagem	*Descrição*
1	Acção de pedido de pesquisa normal.
1->2	Verificação dos dados do *subscriber*: ▪ Se o modo de descoberta está activo; ▪ Se está ligado à internet.
2	**1->2 OK**: Pedido de pesquisa normal, com os dados do *subscriber*. ➔ *JSON { "tag" : "contentsall" , " username" : "nome" , " radius_value" :" valor do raio de descoberta"," texto_on" :" :" activo ou não "," image_on" :" :" activo ou não "," graphics_on" :" :" activo ou não "," preferences_on" :":" activo ou não "," cultural_on" :" :" activo ou não "," economics_on" :" activo ou não ", " sports_on" :" activo ou não ", " education_on" :" activo ou não ", " science_on" :" activo ou não " }*
3	Pedido dos conteúdos que enquadram nos campos da *String JSON* da tabela de *content*.
4	Resposta ao pedido: ➔ **OK**: Sucesso na obtenção de conteúdos. ➔ **Erro**: Erro na obtenção de conteúdos.
5	Resposta ao pedido para o *subscriber*, enviando a resposta na seguinte *String JSON*: ➔ **OK**: Sucesso na obtenção de conteúdos. Exemplo: *{"tag":"contentall","success":1,"error":0, "0":{"contentId":"59","name":"chart example","type":"chart","theme":"education","JSONfile":"Contents\/7ffc4f66d6.aloh.csv","latitude":"38.659870","longitude":"-9.205356","review":"bom !!!"},"1":{mesma estrutura que o "0"}, "2" (se existirem mais conteúdos que se enquadram na pesquisa) }* ➔ **Erro**: Erro na obtenção de conteúdos. *{"tag":"contentall","success":0,"error":1,"error_msg":"User doesn't exist"}* *{"tag":"contentall","success":0,"error":2,"error_msg":"The user doesn't allow the reception of any content type"}* *{"tag":"contentall","success":0,"error":3,"error_msg":"The user doesn't allow the reception of any content theme"}* *{"tag":"contentall","success":0,"error":4,"error_msg":"Content is not active"}* *{"tag":"contentall","success":0,"error":5,"error_msg":"JSON ERROR"}*

3.5.2.7 Pesquisa rápida (pelo nome do conteúdo)

O *subscriber*, após autenticado, pode desejar fazer uma pesquisa dos conteúdos pelo seu nome. Para melhor entender essa opção de pesquisa a Figura 3-25 ilustra o fluxograma com as trocas de pedidos e verificações e a Tabela 31 os seus detalhes.

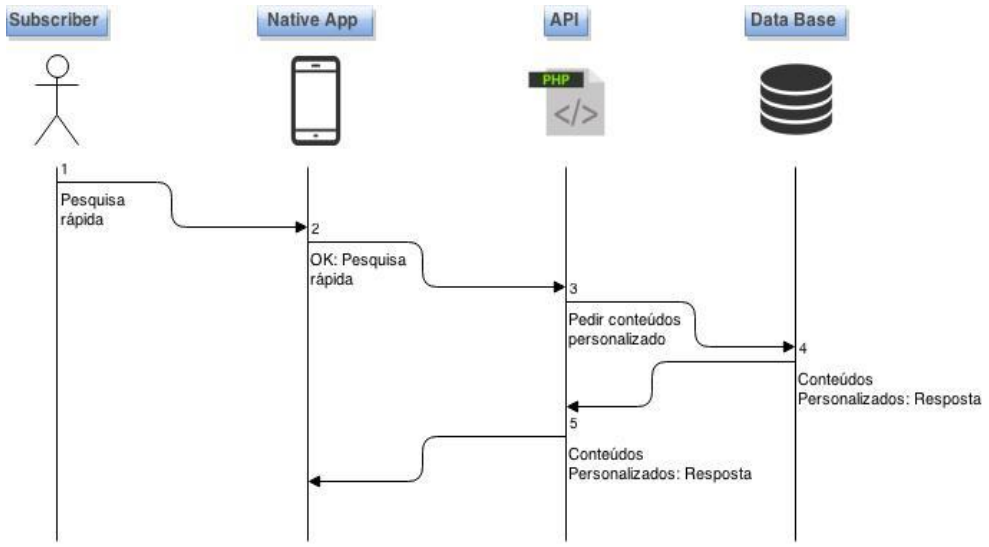


Figura 3-25 Ciclo de mensagens / pedidos entre as entidades e o actor na pesquisa rápida do *subscriber*.

Tabela 31- Descrição mais detalhada das mensagens trocadas entre as entidades e o actor na pesquisa rápida do *subscriber*.

Nº da mensagem	*Descrição*
1	Acção de pedido de pesquisa rápida.
1->2	Verificação dos dados do *subscriber*: ▪ Se todos os campos necessários estão preenchidos; ▪ Se está ligado à internet.
2	**1->2 OK**: Pedido de pesquisa rápida, com os dados do *subscriber*. ➔ *JSON { "tag" : "contentspecific" , " username" : "nome" , " input_content" :" nome do conteúdo para pesquisa" }*
3	Pedido dos conteúdos que enquadram nos campos da *String JSON* da tabela de *content*.
4	Resposta ao pedido: ➔ **OK:** Sucesso na obtenção de conteúdos. ➔ **Erro:** Erro na obtenção de conteúdos.
5	Resposta ao pedido para o *subscriber*, enviando a resposta na seguinte *String JSON*: ➔ **OK:** Sucesso na obtenção de conteúdos. Exemplo: `{"tag":"contentspecific","success":1,"error":0, "0":{"contentId":"59","name":"chart example","type":"chart","theme":"education","JSONfile":"Contents\/7ffc4f66d6.aloh.csv","latitude":"38.659870","longitude":"-9.205356","review":"bom !!!"},"1":{mesma estrutura que o "0"}, "2" (se existirem mais conteúdos que se enquadram na pesquisa) }` ➔ **Erro:** Erro na obtenção de conteúdos. `{"tag":"contentspecific","success":0,"error":1,"error_msg":"User doesn't exist"}` `{"tag":"contentspecific","success":0,"error":2,"error_msg":"There are no contents with that name"}` `{"tag":"contentspecific","success":0,"error":3,"error_msg":"There are no active contents with that name"}` `{"tag":"contentspecific","success":0,"error":5,"error_msg":"JSON ERROR"}`

4. Validação

Este capítulo tem como objectivo validar o modelo proposto. Para isso, são mostradas as condições de funcionamento, como outras verificações não abordadas anteriormente e o uso de bibliotecas externas. O uso de um administrador é validado e testado neste capítulo, bem como alguns extras para melhorar o funcionamento da solução. Para finalizar é demonstrado um teste com dados reais fornecidos pela *IrRADIARE*, onde se demonstra todo potencial desta solução.

4.1. Descrição da solução *Web-based* utilizada pelo *publisher*

A implementação do modelo proposto para a solução *Web-based* utilizada pelo *publisher* descrita no capítulo 3 será aprofundada neste subcapítulo, onde o leitor poderá visualizar a interface com o utilizador (*publisher*), bem como outras funcionalidades, não descritas anteriormente, com mais detalhe.

4.1.1. Publisher Interface

O *publisher* ao abrir a aplicação *Web-based* encontrará o seguinte ambiente ilustrado pela Figura 4-1.

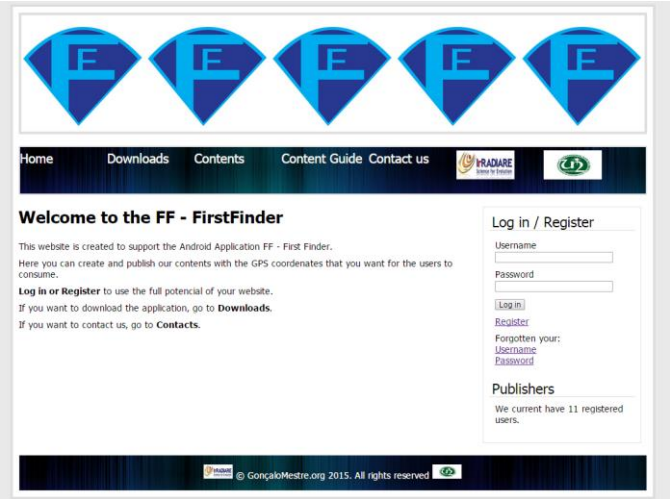


Figura 4-1 Ambiente inicial na abertura da aplicação *Web-based*.

Esta interface é caracterizada por um *banner* com o logotipo da aplicação, uma barra de navegação com diferentes opções de escolha para o utilizador, uma *sidebar* com os *widgets* das opções de *log in* / registo referentes ao perfil do *publisher* e o número de *publishers* inscritos na aplicação, um *footer* com informações sobre os direitos da aplicação e o nome do seu criador e por fim, o corpo que mudará conforme qual o *script* da aplicação está exibido no ecrã.

Destas cinco estruturas, apenas três serão estáticas durante a utilização: o *banner*, a barra de navegação e o *footer*. As duas restantes, *sidebar* e corpo, vão ser dinâmicas, dependendo a sua exibição de certas condições. Por exemplo, a *sidebar* antes do *publisher* fazer o *log in* exibe os *widgets* da Figura 4-1, onde o *publisher* pode inserir o *username* e a *password* para fazer o *log in*, pode ir para a página destino para o seu registo, ou relembrar o seu *username* ou *password*. Para além destas funcionalidades, é exibido o número de *publishers* inscritos na aplicação.

A barra de navegação é composta por várias opções, como a hiperligação *Home* é destinada para a página principal da aplicação. O campo *Downloads* corresponde à hiperligação com página onde é possível fazer o *download* da aplicação para o consumo dos conteúdos, a aplicação móvel. Nesta página estão também as condições necessárias para a sua instalação e como fazê-la.

A hiperligação *Contents* é destinada para mostrar as hiperligações referentes aos conteúdos do *publisher*, estando o seu conteúdo só disponível quando o utilizador se encontra *log in*. O campo *Content Guide* é uma página destinada para explicar as várias extensões de conteúdos aceites na aplicação, e seus respectivos tipos e formulários, encontrando-se também só disponível quando o utilizador está *log in*.

Por fim, a hiperligação *Contact Us* é destinada para fornecer os contactos da equipa deste projecto. Os restantes campos da barra de navegação, são hiperligações para os sites oficiais das duas entidades que apoiaram esta dissertação, a *IrRADIARE* e a Universidade Nova de Lisboa.

Após o *publisher* fazer o *log in* aparecem novos *widgets* na *sidebar*, ilustrados na Figura 4-2.

Figura 4-2 *Widgets* da *sidebar* depois do publisher fazer a autenticação (*log in*).

Começando de cima para baixo, o *publisher* nesta *sidebar* encontra todas as hiperligações e *widgets* para ver e alterar o seu perfil. Ele pode mudar a sua foto de perfil, escolhendo um ficheiro com a extensão: *jpg*, *png*, *gif* ou *jpeg*. Se desejar fazer o *log out*, só precisa de carregar na hiperligação com esse nome. Para além destas opções o *publisher* pode ver o seu perfil carregando na hiperligação *Profile* e alterar alguns dos seus dados carregando na opção *Settings* ou *Change Password* para alterar a sua *password.*

O corpo da página inicial será composto por uma breve explicação da aplicação e sua utilidade, bem como os diferentes títulos da barra de navegação.

4.1.1.1Um novo registo

Um novo *publisher,* para poder usufruir de todo o potencial da aplicação deve-se registar na base de dados da mesma. Para tal, só precisa carregar na hiperligação *Register* situada na *sidebar* para visualizar o formulário de registo, ilustrado na Figura 4-3.

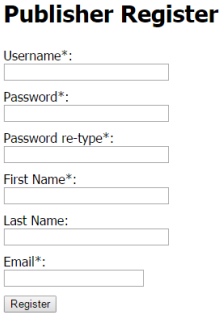


Figura 4-3 Formulário para o registo do *publisher*.

Os campos do formulário com *(asterisco) são de preenchimento obrigatório. Após o novo *publisher* preencher todos os campos necessários, a submissão da informação é feita carregando no botão *Register* do final do formulário.

Para o *publisher* ser inserido na base de dados deverá:

- Ter preenchido todos os campos obrigatórios;
- Ter um *username* sem espaços;
- A *password* ter entre 6 a 32 caracteres;
- Ter um *email* válido;
- Não ter os campos *email* e *username* igual a nenhum *publisher* que exista na base de dados.

Após o utilizador ser inserido na base de dados, um *email* com o código de activação é enviado para o *publisher* poder activar a sua conta. Só depois da activação é que ele pode fazer o *log in* com sucesso.

4.1.1.2 Recuperação da *password* ou *username*

Após o registo, o utilizador pode-se esquecer da sua *password* ou do seu *username.* Para recuperá-los só necessita de carregar na hiperligação *Username* ou *Password* localizadas por baixo do texto *"Forgotten your:"*. O formulário encontra-se ilustrado na Figura 4-4.

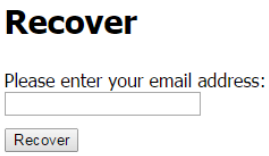


Figura 4-4 Formulário para a recuperação da *password* ou *username* do *publisher*.

O utilizador insere o *email* do *publisher* que deseja recuperar o *username* ou a *password* dependendo do que este deseja recuperar. Se o *email* for válido e se o encontrar na base de dados, um *email* com a informação pedida será enviado para o *email* da base de dados. Se for o esquecimento de uma *password*, uma nova será atribuída ao *publisher* e enviada para o *email*.

4.1.1.3 Ver e modificar o seu perfil e *password*

Após o *publisher* fazer o *login* este pode ver ou modificar o seu perfil. Na *sidebar* terá três opções para escolher: *Profile*, *Change Password* e *Settings*.

A opção *Profile*, mostra os dados do perfil utilizador.

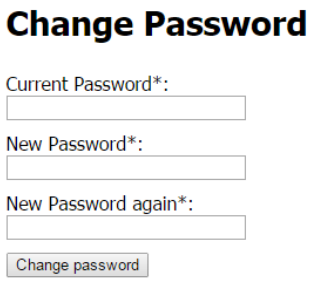


Figura 4-5 Formulário para a mudança de *password* do *publisher*.

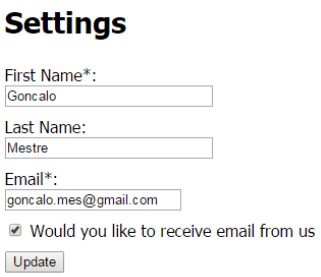


Figura 4-6 Formulário para a mudança de dados do *publisher*.

A opção *Change Password*, permite ao utilizador mudar a sua *password*, é ilustrada no formulário da Figura 4-5. O *publisher* terá de colocar a *password* actual correcta e a nova *password* deverá conter entre 6 a 32 caracteres para obter com sucesso uma mudança de *password*.

A opção *Settings* permite modificar alguns dos dados do perfil do *publisher*, ilustrado no formulário da Figura 4-6. Para mudar os dados da base de dados o utilizador deve preencher todos os campos obrigatórios, ter um *email* válido e não existir nenhum *email* igual já guardado na base de dados. O campo *"Would you like to receive email from us"* permite ao utilizador dar a escolha de receber ou não *emails* ao utilizador. Este campo é destinado ao envio de *emails* com as novidades da aplicação.

4.1.1.4Ver guia de conteúdos

Após o *publisher* fazer o *log in* é possível ver o guia dos conteúdos aceites para publicação na aplicação, carregando no campo *Content Guide* da barra de navegação. A Figura 4-7 ilustra o guia.

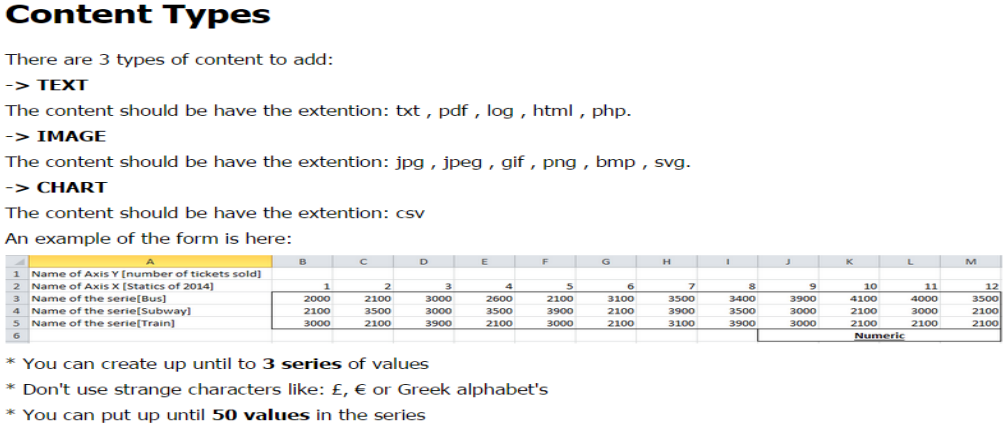


Figura 4-7 Corpo da página do guia dos conteúdos (*Content Guide*).

Existem três tipos de conteúdos: texto, imagem e gráfico. Para os conteúdos de texto são aceites as extensões: *txt, pdf, log, html* e *php*. Para os conteúdos de imagem são aceites as extensões: *jpg, jpeg, gif, png, bmp* e *svg*. Para os conteúdos de gráfico só são aceites no formato: *csv*, porém este tipo de conteúdos ainda tem outra verificação para a correcta visualização dos gráficos na aplicação móvel:

- Verifica se o ficheiro se encontra parecido ao do exemplo do guia;
- Se os valores do eixo do X não são duplicados;
- Se tem menos de 50 valores nas suas séries;
- Se tem colunas incompletas;
- Se tem mais linhas que o normal (Máximo 5 linhas);´
- Verifica o uso de caracteres malignos, como $, € ou do alfabeto grego;
- Se todos os valores inseridos nas séries são numéricos.

4.1.1.5 Adicionar um novo conteúdo

O *publisher* para adicionar um novo conteúdo deverá escolher o campo *Contents* da barra de navegação, disponível só após o *log in* do utilizador. Ao carregar nesta hiperligação, aparece a seguinte opção de escolha:

Figura 4-8 Corpo da escolha do campo *Contents* da barra de navegação.

A partir deste *script* o utilizador pode escolher adicionar um novo conteúdo ou visualizar todos os seus conteúdos. Este subcapítulo foca-se no adicionar de novos conteúdos, então escolhe-se a hiperligação de *Add a new content* para adicionar um novo conteúdo, ilustrando o seguinte formulário da Figura 4-9 após escolhida esta opção.

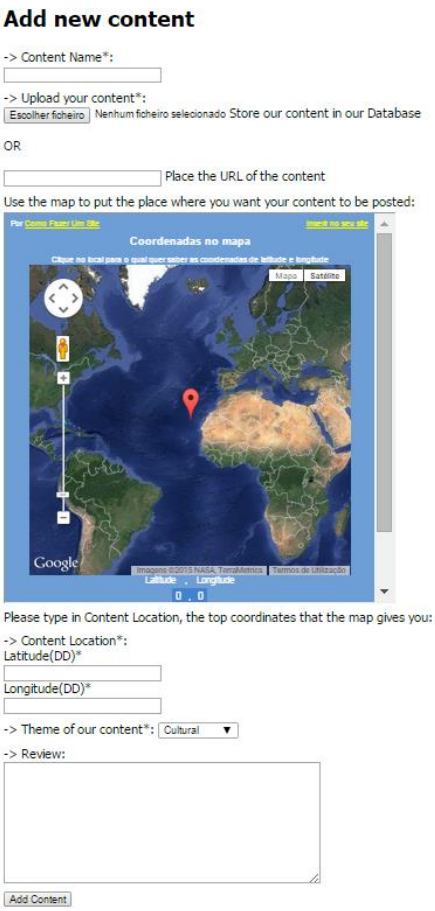


Figura 4-9 Formulário para adicionar um novo conteúdo.

A partir do formulário da Figura 4-9 o adicionar de um novo conteúdo é possível desde que:

- O utilizador preencha todos os campos obrigatórios (assinalados com *);
- Não existir nenhum conteúdo com um nome igual na base de dados;
- Escolher um ficheiro / *URL* válido;
- O ficheiro / *URL* ter uma extensão ou formato válida;
- O conteúdo ser inferior a 1Mb;
- Se o conteúdo for um gráfico, deverá passar por todas as verificações abordadas no *Content Guide*;
- Se o *ficheiro* / *URL* já exista.

O conteúdo pode ser de um dos seguintes temas: cultural, educacional, económico, desportivo ou científico. Se estas condições forem alcançadas o conteúdo será adicionado com sucesso na base de dados.

4.1.1.6 Visualizar, editar ou remover conteúdos

A partir do *script* da Figura 4-8 escolhe-se a hiperligação *View our contents* para visualizar os conteúdos que o *publisher* tem na base de dados. A Figura 4-10 ilustra a visualização dos conteúdos do *publisher* autenticado.

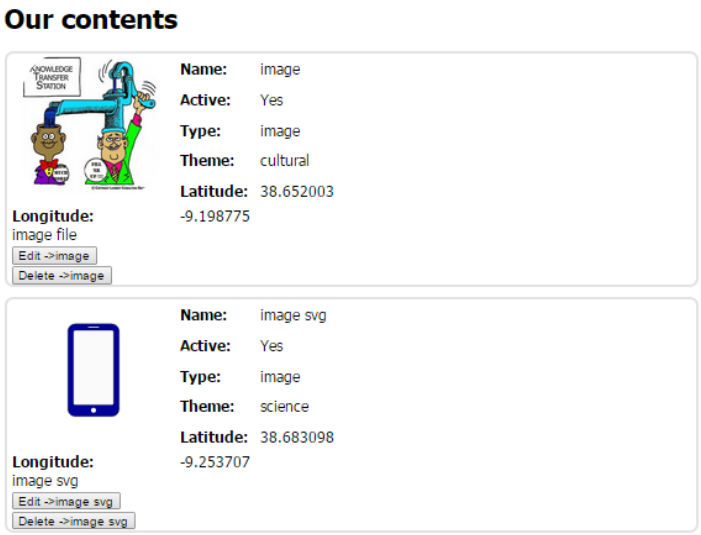


Figura 4-10 Formulário da visualização dos conteúdos do *publisher*.

A partir da Figura 4-10, pode-se ver a lista dos conteúdos do *publisher*, onde para cada um, é apresentado o seu nome, se se encontra activo, qual o seu tipo e tema, as suas coordenadas geográficas e a sua revisão. Se o conteúdo for do tipo imagem, a sua imagem aparecerá nos detalhes, se não for aparecerá uma imagem predefinida para os conteúdos de texto e gráfico.

A partir desta visualização o *publisher* pode escolher editar ou eliminar o conteúdo, carregando no botão *edit->"nome do conteúdo"* ou *delete->"nome do conteúdo"*. Se o utilizador escolher editar o conteúdo o formulário da Figura 4-11 necessitará de ser preenchido.

Figura 4-11 Formulário referente à edição de um conteúdo do *publisher*.

Ao abrir o formulário os vários campos são preenchidos com os dados guardados na base de dados, excepto o escolher o ficheiro ou *URL*. Assim o utilizador pode actualizar os dados se o desejar. Após o conteúdo ser inserido, por predefinição o conteúdo fica automaticamente activo, é neste formulário que o *publisher* pode desejar colocar o seu conteúdo inactivo, não permitindo o seu consumo por parte do *subscriber* na aplicação móvel, mas não está eliminado da base de dados. Todas as verificações referidas anteriormente (na adição de um novo conteúdo e as verificações para os conteúdos do tipo gráfico), aplicam-se neste passo.

Se o utilizador desejar eliminar o conteúdo só precisa de carregar no botão *delete->"nome do conteúdo"* do formulário da Figura 4-10.

4.1.2. Administrador dos *Publishers*

Para facilitar a gestão dos publishers existe um *publisher* administrador da aplicação. Este administrador interage na mesma interface que os *publishers* normais e pode fazer todas as acções descritas anteriormente, porém tem alguns privilégios para administrar os outros utilizadores.

Alguns destes privilégios são mostrados na página inicial se o *publisher* for o administrador. A Figura 4-12 mostra o corpo da página principal do administrador.

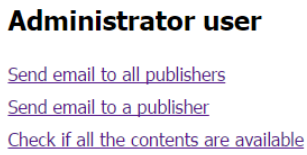


Figura 4-12 Corpo da página inicial do administrador dos *publishers*.

A partir da Figura 4-12 podem-se ver três hiperligações a mais, face ao utilizador normal. O *Send email to all publishers* é utilizado para fazer a hiperligação para o formulário do envio de uma mensagem para todos os utilizadores. Este formulário é ilustrado na Figura 4-13.

Figura 4-13 Formulário do administrador dos *publishers* para enviar um *email* para todos.

Neste formulário, o administrador só precisa de preencher o *subject* e o corpo da mensagem e carregar no botão *SEND*, para enviar uma mensagem para todos os *publishers* da base de dados. Este tipo de formulário pode ser muito interessante para enviar actualizações e novidades para todos os utilizadores.

O *Send email to a publisher* é utilizado para fazer a hiperligação para o formulário do envio de uma mensagem para um *publisher*. Este formulário é ilustrado na Figura 4-14.

Figura 4-14 Formulário do administrador dos *publishers* para enviar um *email* para um *publisher*.

Neste formulário o administrador só precisa de preencher o *email* com o *email* do *publisher* que deseja enviar a mensagem, o *subject* e o corpo da mensagem e carregar no botão *SEND*, para enviar a mensagem para o *publisher* escolhido, se ele se encontrar na base de dados. Este tipo de formulário é muito interessante para enviar mensagens privadas de avisos a um *publisher* com comportamento reprovável.

O *Check if all the contents are available* é utilizado para fazer a hiperligação para o *script* que verifica se todos os conteúdos estão válidos, ou seja, se é possível aceder ao seu *URL*. Se algum dos *URLs* dos conteúdos na base de dados não for possível de aceder, um display será feito no corpo da página do *script* anunciando qual é o conteúdo que não é válido, colocando o conteúdo inactivo e enviando automaticamente um *email* para o *publisher* a que pertence o conteúdo descrevendo o problema.

Esta verificação é muito interessante para conteúdos cujo *URL* já não existe e estão a aparecer como activos para o *subscriber* consumir.

Outro privilégio do administrador é o de poder visualizar todos os conteúdos dos *publishers.* Para tal o administrador só precisa de carregar no campo *Contents* da barra de navegação, onde poderá encontrar a hiperligação para a visualização de todos os conteúdos (*View all contents*).

A Figura 4-15 ilustra a visualização de todos os conteúdos da base de dados após a hiperligação *View all contents*, ser escolhida.

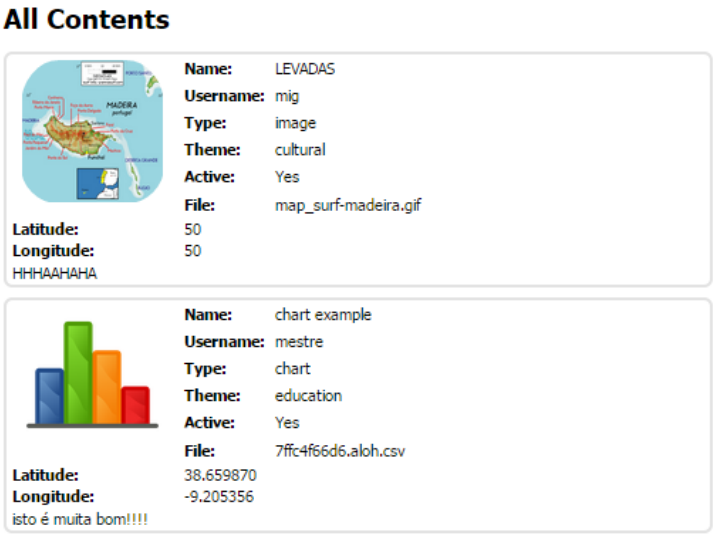


Figura 4-15 Visualização de todos os conteúdos da base de dados.

A partir da Figura 4-15 observam-se vários detalhes sobre os conteúdos, como o seu nome, o *publisher* a que pertence, o seu tipo e tema, se está activo ou não, o nome do ficheiro, as suas coordenadas geográficas e a revisão do *publisher*. Este tipo de visualização permite ao administrador saber a quem pertence o conteúdo e nome do ficheiro onde este está guardado, coisa que o *publisher* normal não consegue.

Todos estes *scripts* utilizados pelo administrador estão protegidos para só serem usados pelo *publisher* do tipo administrador, negando o acesso aos restantes *publishers*.

4.2. Descrição da solução *Navite-app* utilizada pelo *subscriber*

A implementação do modelo proposto para a solução *Native-app* utilizada pelo *subscriber* descrita no capítulo 3 será aprofundada neste subcapítulo, onde o leitor poderá visualizar, com mais detalhe, a interface com o utilizador (*subscriber*), bem como outras funcionalidades não descritas anteriormente com mais detalhe.

4.2.1. *Subscriber* Interface

O *subscriber* ao abrir a aplicação móvel *Native-app Android* encontrará o seguinte ambiente ilustrado pela Figura 4-16.

Figura 4-16 Ambiente inicial na abertura da interface da *Native-app Android* do *subscriber*.

Esta interface é caracterizada por usar o logotipo da aplicação no canto superior esquerdo, juntamente com o seu nome durante o seu uso. Esta parte da estrutura da aplicação manter-se-á estática durante o seu uso. Por baixo, o utilizador encontra uma pequena informação sobre o uso da aplicação e os campos para se autenticar. Continuando a seguir na mesma direcção, se o *subscriber* se esquecer de algum dos dados de autenticação para o *Log in*, pode carregar na frase *Forgot your Username* ou *Forgot your Password*, para os recuperar. Os dois botões no final são usados para abrir as interfaces do registo do *subscriber* (*Register*) e para ir para a aplicação *Web-based* dos *publishers* (Botão com o mundo).

4.2.1.1 Um novo registo

Um novo *subscriber* para poder fazer o *log in* e usufruir de todo o potencial da aplicação, deve-se registar na base de dados. Para tal, o novo *subscriber* só precisa de carregar no botão *Register* para abrir a interface de registo, ilustrada na Figura 4-17.

Figura 4-17 Formulário para o registo de um novo *subscriber* na base de dados.

Todos os campos do formulário devem ser preenchidos para o registo ser completo. Porém, também existem algumas verificações a serem feitas antes de adicionar um novo *subscriber* na base de dados.

Para o *subscriber* ser inserido na base de dados deverá:

- Ter preenchido todos os campos do formulário;
- Ter *username* maior que 4 caracteres;
- A *password* ter no mínimo 6 caracteres;
- Os campos *password* e *password re-type* devem ser iguais;
- Estar ligado à internet;
- O *username* já se encontrar na base de dados;
- O *email* já se encontrar na base de dados.

Após o *subscriber* ser inserido na base de dados, são colocados os valores predefinidos nas tabelas *discovery* e *preferences* relativos ao *subscriber* recém registrado e é enviado um *email* que é encaminhado para a interface de visualização dos seus dados de registo, ilustrada na Figura 4-18.

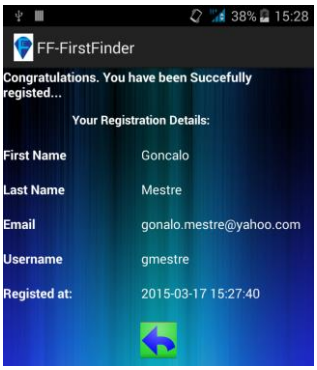


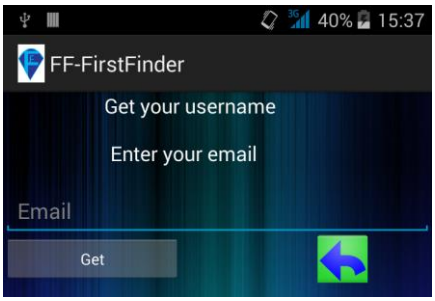
Figura 4-18 Interface para visualização dos dados do registo com sucesso do *subscriber*.

A partir da Figura 4-18 observa-se os detalhes do registo do *subscriber* na base de dados, o 1º nome, o sobrenome, o *email*, o *username* e a data e hora a que foi registado.

4.2.1.2 Recuperação da *password* ou *username*

Após o registo, o *subscriber* pode-se esquecer da sua *password* ou do seu *username.* Para recuperá-los só necessita de carregar na frase *Forgot your Username* ou *Forgot your Password* da interface da Figura 4-19. O formulário da recuperação do *username* está ilustrada na Figura 4-20.

Figura 4-19 Formulário para a recuperação da a *password* do *subscriber* na base de dados.



4-20 Formulário para a recuperação do *username* do *subscriber* na base de dados.

No caso do esquecimento do *username* o utilizador preenche o formulário com o seu *email*, e se o *email* existir na base de dados do *subscriber* e estiver ligado à internet, um *email* será enviado para o *subscriber* com o *username* do *email* preenchido no formulário.

No caso do esquecimento da *password*, o utilizador preenche o formulário com o seu *username*, e se o *username* existir na base de dados do *subscriber* e estiver ligado à internet, um *email* será enviado para o *subscriber* com a uma nova *password* aleatória para o *username* preenchido no formulário.

4.2.1.3 *Log in* (Autenticação)

O *subscriber*, após registar-se pode fazer o *log in* para ir usufruir de todo o potencial da aplicação. Para tal, só preciso de preencher os campos de autenticação do formulário da Figura 4-16, estar ligado à internet e carregar no botão verde *Login*.

Se a combinação *username* e *password* estiver correcta com a guardada na base de dados, o *subscriber* avança para a interface principal da aplicação, ilustrada na Figura 4-21 após a autenticação estar feita. Para além desta mudança de interface também são transferidos alguns dos campos das tabelas *subscriber*, *discovery* e *preferences* da base de dados para a tabela interna *login* da aplicação móvel.

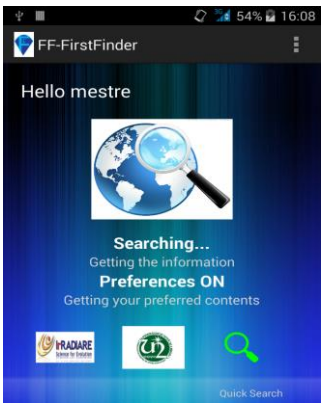


Figura 4-21 Interface principal após o *subscriber* fazer o *log in*.

Esta interface é o ponto de partida para pesquisar e consumir os conteúdos. Observando a sua estrutura do topo para baixo, tem-se a opção de ver o menu do perfil do *subscriber* nos três pontos posicionados na vertical do canto superior direito, saúda-se o *subscriber* autenticado, informa-se o utilizador se a aplicação se encontra a pesquisar (*Searching…*) ou não (*Discorey Off*) os conteúdos dentro do raio predefinido pelo *subscriber*, seguido de uma pequena informação sobre a pesquisa, *Getting the information* se o modo de descoberta estiver activo ou *If you want to get the contents, please turn the discovery mode on* para quando o modo de descoberta está desactivado, informando o utilizador de como o colocar activo.

Em seguida, o subscriber é avisado se só recebe ou não os conteúdos segundo as suas preferências (*Preferences ON* ou *OFF*), seguido de uma pequena informação sobre a obtenção de conteúdos, *Getting your preferred contentes*, para quando o modo de preferência está activo e *Getting all the contents* para quando está inactivo.

Por fim encontram-se três botões para o utilizador carregar se desejar. O da *IrRADIARE* (se for carregado abre o site oficial da empresa *IrRADIARE*), o da *UNL* (se for carregado abre o site oficial da Universidade Nova de Lisboa) e por fim a lupa, que tal como a descrição abaixo explica é utilizado para navegar para a interface da pesquisa rápida (pesquisa pelo nome do conteúdo).

Quando o *subscriber* se encontra nesta interface, não precisa carregar em nenhum botão para procurar os conteúdos próximos de si (pelo raio), pois no *background* a aplicação já está a fazer isso se o modo de descoberta estiver activo.

4.2.1.4 Menu de perfil do utilizador

Na interface da Figura 4-21 o utilizador pode carregar nos três pontos posicionados na vertical do canto superior direito, para fazer aparecer o menu de personalização da pesquisa e perfil do *subscriber*. Ao fazer esta acção aparecerá o menu ilustrado na Figura 4-22.

Figura 4-22 Menu de personalização da pesquisa e perfil do *subscriber*.

O *subscriber* pode escolher uma das seis opções:

- *Discovery Preferences:* esta opção é escolhida para modificar as preferências da descoberta. A interface desta opção é mostrada na Figura 4-23.

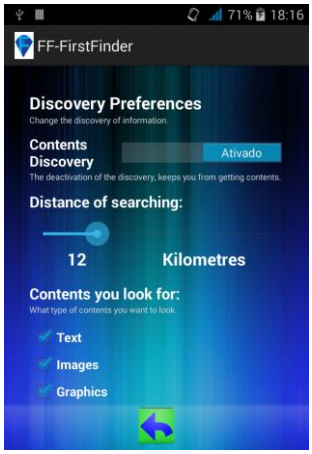


Figura 4-23 Interface das escolhas da preferência da descoberta do *subscriber*.

Nesta interface coloca-se activa ou não a descoberta de conteúdos, a distância máxima que o utilizador deseja procurar conteúdos em quilómetros e quais os tipos de conteúdos que deseja receber. Todas as modificações feitas nesta interface vão afectar a tabela interna *login,* que por sua vez vão afectar a pesquisa do *subscriber*.

- *Content Settings:* esta opção é escolhida para modificar as preferências dos conteúdos descobertos. A interface desta opção é mostrada na Figura 4-24.

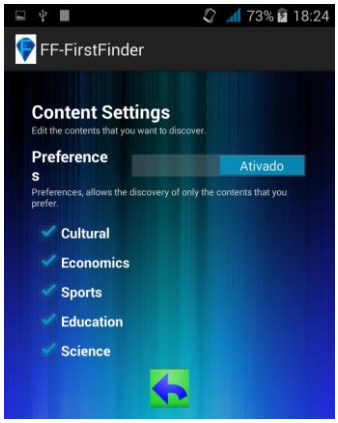


Figura 4-24 Interface das escolhas da preferência dos conteúdos descobertos do *subscriber*.

Nesta interface coloca-se activa ou não a preferência dos temas de conteúdos a serem descobertos, podendo o *subscriber* escolher quais os temas que deseja procurar. Todas as modificações feitas nesta interface vão afectar a tabela interna *login,* que por sua vez vão afectar a pesquisa do *subscriber*.

- *Our profile:* esta opção é escolhida para ver e modificar o perfil do *subscriber*. As interfaces desta opção são mostradas nas Figura 4-25 e Figura 4-26.

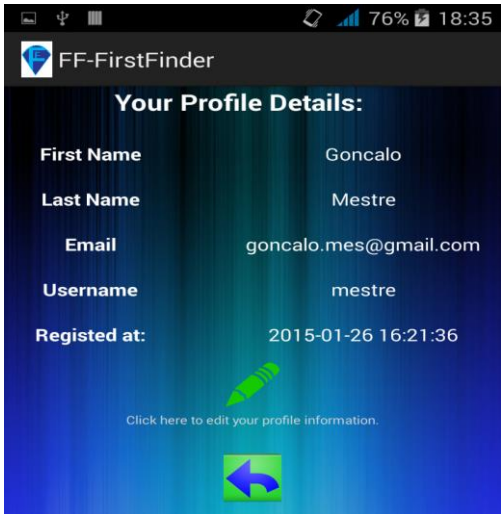


Figura 4-25 Interface para visualizar o perfil do *subscriber*.

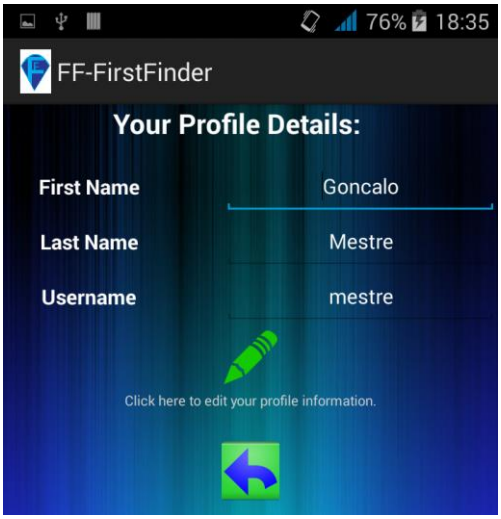


Figura 4-26 Interface modificar o perfil do *subscriber*.

Na Figura 4-25 visualiza-se o perfil, onde se encontram os vários dados do *subscriber* como o 1º nome, o sobrenome, o *email*, o *username* e a data e hora a que foi registado. Por fim, o utilizador pode carregar no botão lápis verde para editar o seu perfil, abrindo uma nova interface mostrada na Figura 4-26.

A modificação do perfil é feita nesta interface, onde o *subscriber* pode alterar os campos do 1º nome, o sobrenome e o *username*. Depois de alterados se desejar modificar é só voltar a carregar no lápis verde outra vez para guardá-los. Caso não deseje fazer nenhuma alteração é só carregar na seta para trás.

- *Change Password:* esta opção é escolhida para modificar a *password* do *subscriber*. A interface desta opção é mostrada na Figura 4-27.

Figura 4-27 Interface para modificar a *password* do *subscriber*.

A mudança da *password* do *subscriber* é feita nesta interface onde após preencher o formulário com a nova *password* é feita uma verificação se a *password* é maior que 5 caracteres, se o *subscriber* existe na base de dados e se está ligado a internet. Se estas condições forem alcançadas, a mudança de *password* é feita com sucesso.

- *Contact Us:* esta opção é escolhida para entrar em contacto com a equipa do *FF – First Finder*. A interface desta opção é mostrada na Figura 4-28.

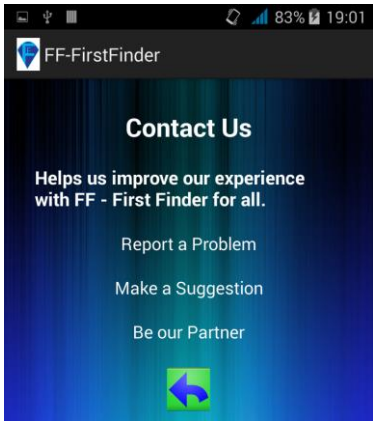


Figura 4-28 Interface para entrar em contacto com a equipa do *FF – First Finder.*

Com esta interface o *subscriber* pode enviar um *email* para a equipa do *FF – First Finder,* com três diferentes *subjects*: reportar um problema, fazer uma sugestão ou tornar-se parceiro. Após o utilizador escolher uma das opções, abrirá o seu *email* e a mensagem está já composta só faltando escrever o que necessita. No topo da mensagem está a versão do sistema operativo *Android* utilizado pelo utilizador.

- *Logout:* esta opção é escolhida quando o subscriber já não deseja procurar mais conteúdos e quer voltar a página inicial da aplicação.

Neste caso, os dados guardados na tabela *login* da aplicação são enviados para a base de dados, actualizando os valores referentes aos mesmos nas tabelas *subscriber*, *discovery* e *preferences*, se o *subscriber* estiver ligado à internet. Caso contrário, todas as modificações da tabela *login* durante o uso do *subscriber* não vão ser guardadas na base de dados, mantendo as definições iguais ao seu último envio.

4.2.1.5 Pesquisa normal

Depois do *subscriber* entrar na interface principal após o *log in* inicia-se a pesquisa normal se o utilizador não carregar em nenhum item relevante durante dez segundos. Após este tempo um pedido de pesquisa de conteúdos, personalizado à maneira do *subscriber*, é enviado para a *API.*

A *API* verifica as seguintes condições do pedido:

- Se o *subscriber* não existe na base de dados;
- Se o *subscriber* não permite a pesquisa de nenhum tipo de conteúdo (texto, imagem ou gráfico);

- Se o *subscriber* não permite a pesquisa de nenhum tema de conteúdo (cultural, económico, educativo, desportivo ou científico);
- Se não existe nenhum conteúdo activo com aquela preferência.

Se estas condições forem alcançadas a pesquisa de conteúdos foi um sucesso e a aplicação encaminha o *subscriber* para a interface da Figura 4-29, onde mostra a lista de conteúdos descobertos com as suas preferências. Esta lista está limitada em oitenta conteúdos, sendo ordenados do mais recente para o mais antigo.

Figura 4-29 Interface para os mostrar a lista dos conteúdos recebidos pela pesquisa normal do *subscriber*.

Esta interface é utilizada mostrar a lista de conteúdos da pesquisa personalizada efectuada pelo *subscriber* a partir da sua posição. Observando a interface verifica-se o número de resultados total da pesquisa e a exibição em forma de lista dos conteúdos encontrados, sendo que em cada exibição de conteúdo vê-se o seu nome, tema, distância a que se encontra do utilizador (sendo actualizada periodicamente), a revisão do *publisher* sobre o seu conteúdo, o seu tipo, imagem, texto ou gráfico (representado pelo botão à esquerda da informação sobre o conteúdo) e a opção de ver onde se encontra este conteúdo no mapa (botão à direita da informação do conteúdo, o globo).

Se o *subscriber* carregar no botão do tipo do conteúdo (o da esquerda) ele estará a escolher consumir o conteúdo, ou seja, é aberto o *URL* do conteúdo através do *browser* para o seu consumo (no caso do tipo ser texto ou imagem) e é colocado como *background* na aplicação uma interface com os detalhes do conteúdo e apoio ao seu consumo. A Figura 4-30 ilustra um exemplo do consumo de um conteúdo do tipo imagem ou texto.

Figura 4-30 Consumo de um conteúdo do tipo imagem.

Esta é a visualização do conteúdo de texto ou imagem, se o *subscriber*, após carregar no botão do consumo, estiver ligado à internet. Caso contrário, a actividade de *background* com os detalhes do conteúdo e apoio ao seu consumo é sobreposta sobre esta visualização. O detalhe desta interface ilustra-se na Figura 4-31.

Figura 4-31 Interface de *background* para apoio ao consumo do conteúdo da aplicação.

Nesta interface o *subscriber* pode voltar a tentar consumir o conteúdo carregando na imagem do seu tipo de novo, ou então ver os detalhes do conteúdo com mais atenção. Se não estiver ligado à internet, o *subscriber* é avisado com uma mensagem por baixo da seta para voltar para a lista. Se o *subscriber* desejar voltar à lista recebida, só precisa de carregar no botão da seta para voltar atrás.

Porém no caso do consumo de um gráfico, o funcionamento do consumo do conteúdo será ligeiramente diferente. O encargo da abertura do conteúdo não fica pelo *browser* mas sim pela aplicação que usa a interface de apoio ao consumo da Figura 4-31 como suporte para abrir, ler e interpretar o formato *csv* do *URL*. Se o *subscriber* estiver ligado à internet, esse formato é aberto a partir do seu *URL*, depois é lido e interpretado pela aplicação segundo as verificações já mencionadas anteriormente para os conteúdos de gráficos. Durante este processo o *subscriber* visualiza as informações mostradas na Figura 4-32 da aplicação.

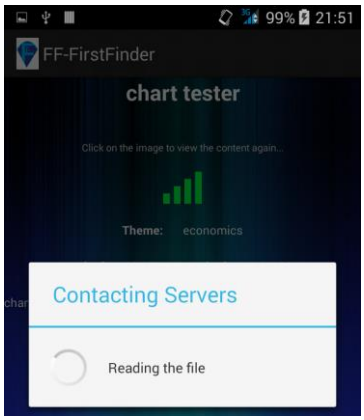


Figura 4-32 Visualização da leitura do formato *csv* pelo *subscriber*.

Se existir algum erro durante este processo, será apresentada uma mensagem ao *subscriber* com a impressão do mesmo (isto incluí o erro de não estar ligado à internet). Se tudo correr bem na leitura e verificação do ficheiro, usa-se a biblioteca *achartengine-1.1.0.jar*[67] para ajudar a exibir os dados do ficheiro sobre a forma de gráfico no ecrã do *subscriber*. A Figura 4-33 mostra o resultado final do uso desta biblioteca.

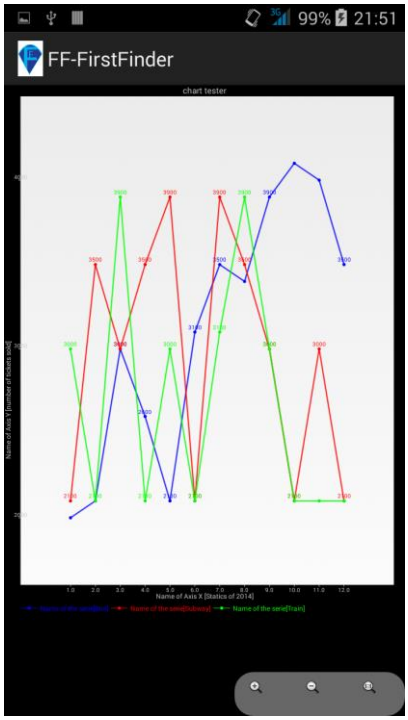


Figura 4-33 Visualização dos dados do ficheiro lido, utilizando a biblioteca *achartengine-1.1.0.jar*.

Este gráfico é interactivo com o utilizador, pois permite fazer os *zooms* necessários, imprime no topo o nome do conteúdo, coloca os nomes nos eixos dos X e Y desejados, permite a visualização das várias séries de dados com várias cores em gráficos lineares e os valores dos seus pontos serem impressos no ecrã.

Se o *subscriber* desejar voltar consumir o gráfico outra vez só precisa de carregar na imagem do tipo da Figura 4-31 para este processo repetir-se outra vez.

Se na interface da Figura 4-29 o *subscriber* carregar no botão com o globo de um conteúdo, o *subscriber* deseja ver onde se encontra esse conteúdo no mapa. Assim é necessário chamar a biblioteca do *Google* para utilizar o *Google Maps* na aplicação, conforme foi abordado no subcapítulo 2.6.1. A visualização dos conteúdos no mapa é mostrada pela Figura 4-34.

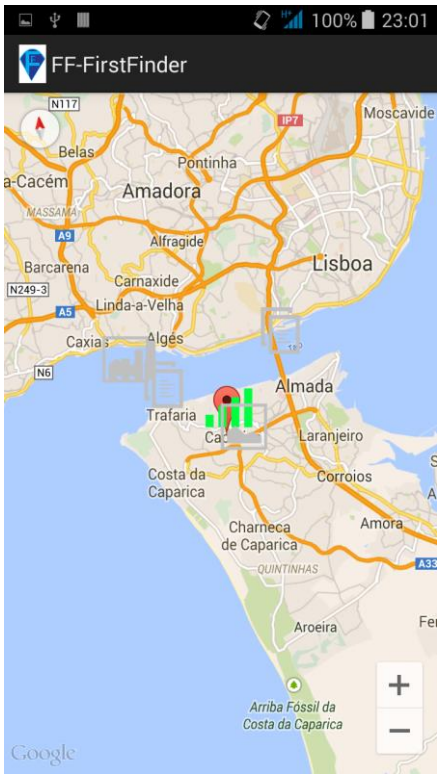


Figura 4-34 Visualização da posição dos conteúdos em redor da posição do *subscriber*.

A partir do mapa da Figura 4-34 verifica-se que o conteúdo escolhido para ser visualizado no mapa se encontra com o *icon* do seu tipo a verde, os restantes encontram-se a cinzento. Se o *subscriber* desejar saber as suas coordenadas basta carregar no *icon* da posição. O mapa actualiza a posição do *subscriber* periodicamente.

O *subscriber* pode consumir qualquer conteúdo no mapa, carregado no seu *icon*, independentemente de ser o conteúdo escolhido ou não.

4.2.1.6 Pesquisa rápida

A seguir ao *subscriber* entrar na interface principal após o *log in*, este pode escolher fazer uma pesquisa rápida carregando no botão da lupa a verde ilustrado na Figura 4-21. Esta acção abre a interface de pesquisa rápida, ou seja pesquisa pelo nome do conteúdo mostrada na interface da Figura 4-35.

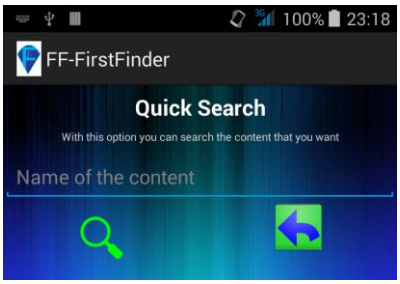


Figura 4-35 Interface de preenchimento para a pesquisa rápida.

O *subscriber* preenche o campo do nome do conteúdo que deseja receber e se o telemóvel estiver ligado a internet envia o pedido para a *API* de obter os conteúdos com esse nome ou similares.

A *API* verifica as seguintes condições do pedido:

- Se o *subscriber* não existe na base de dados;
- Se não existem conteúdos com esse nome;
- Se não existem nenhuns conteúdos activos com esse nome.

Se nenhuma destas condições existir, significa que foi encontrado pelo menos um conteúdo com esse nome ou similar. A Figura 4-36 mostra um exemplo de uma pesquisa deste tipo, com a visualização da lista de conteúdos recebida. Esta lista foi limitada em 20 conteúdos e ordenada no mais recente para o mais antigo.

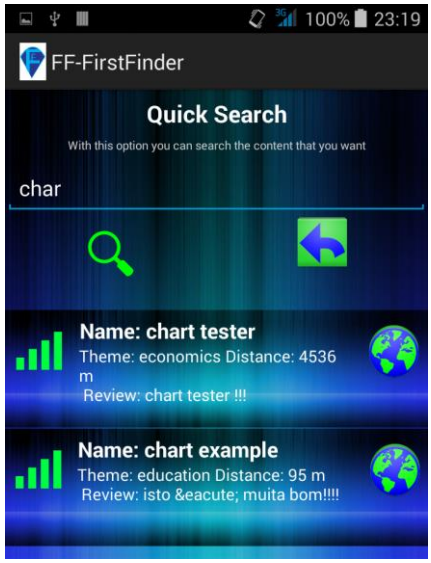


Figura 4-36 Exemplo de uma pesquisa rápida pelo nome '*char*' e seus resultados.

A partir do exemplo da Figura 4-36, verifica-se a lista de conteúdos com o nome '*char*' ou similar. O consumo dos conteúdos e a sua visualização no mapa é idêntico ao descrito anteriormente no subcapítulo 4.2.1.5.

4.3. Teste de validação de criação de gráficos em tempo real

Neste subcapítulo é validado a criação de gráficos a partir de dados em tempo real. Este teste é particularmente interessante para visualizar que estão a ser actualizados constantemente, variando o seu *output* conforme as horas e os dias da semana. Para tal, a *IrRADIARE* forneceu vários *URLs* de dados reais com o formato correcto para o consumo da aplicação móvel. Estes dados são localizados na zona do Norte e actualizados de hora a hora. Após estes conteúdos serem adicionados à base de dados, o seu consumo por parte do *subscriber* é feito a partir do *URL*. Assim, consoante a data e hora estes valores alteram-se, como demonstrado na Figura 4-37.

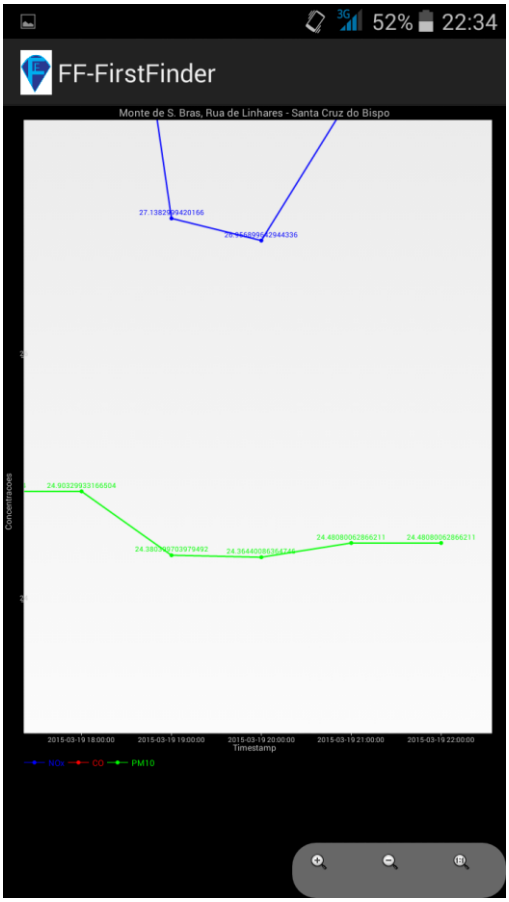


Figura 4-37 Gráfico de dados reais, actualizado de hora a hora.

Observando a Figura 4-37 pode-se notar que os dados visualizados são referentes às últimas horas, pois o telemóvel marca 22:34 e o dado mais recente é datado das 22h do mesmo dia.

5. Conclusões e perspectivas futuras

Esta dissertação foi desenvolvida no âmbito do desenvolvimento de uma solução para um cliente *mobile* para consumo e disponibilização de serviços georreferenciados. Em seguida é feita uma reflexão sobre o trabalho e os resultados obtidos. Para além disso são propostos novos caminhos a seguir.

5.1. Conclusão

As aplicações móveis vieram para ficar. Cada vez há mais *smartphones* a serem vendidos por todo o mundo e os telemóveis com o *OS Android* continuam a ter uma grande procura da parte dos utilizadores. Assim, é normal existir um desenvolvimento de novas aplicações móveis e *APIs* relacionadas com esta área com o objectivo de satisfazer as necessidades dos consumidores.

A proposta apresentada nesta dissertação pretende ser uma mais-valia entre as aplicações móveis e as *APIs* do mercado. Esta proposta cria duas novas ferramentas para o utilizador de *smartphone*: uma aplicação *Web-based* para o utilizador se poder registar e publicar os seus conteúdos georreferenciados e uma aplicação *Native-App Android* para o utilizador consumir os conteúdos publicados. A comunicação entre estas entidades é feita através de uma *API* que controla e regula a publicação e o consumo de conteúdos e insere os dados necessários na base de dados.

A utilização de uma aplicação *Web-based* para a publicação de conteúdos é importante pois o seu uso permite a publicação de conteúdos de vários *OS* diferentes sem estarem restringidos a um computador, *smartphone* ou *tablet*.

A utilização de bibliotecas externas nas aplicações desenvolvidas, permite uma maior interacção com o utilizador, facilitando a interpretação e visualização dos dados bem como o consumo dos conteúdos, beneficiando o utilizador final com a sua simplicidade.

5.2. Trabalhos futuros

Como trabalhos futuros há pequenos aspectos que deverão ser melhorados.

Um ponto a resolver seria a possibilidade de colocar vários conteúdos ao mesmo tempo a partir da recepção de uma *array*. Isto é, como a aplicação *Web-based* foi desenhada, o *publisher* teria de adicionar conteúdo a conteúdo na base de dados, o que pode ser um problema para quando um *publisher* deseja publicar vinte ou trinta conteúdos para aplicação. A existência de um campo preparado para receber uma *array* com um certo formato no seu formulário seria uma melhoria a pensar no futuro.

Outro ponto interessante, seria a possibilidade de colocar os conteúdos activos só durante um período de tempo. Esta funcionalidade seria ideal para campanhas publicitárias ou para conteúdos que são mais desejados durante certas alturas do ano. O *publisher* não teria de se preocupar em activar e desactivar o conteúdo, pois a *API* faria isso automaticamente.

A segurança da *password* do utilizador poderá ser um dos pontos a reforçar. A utilização da encriptação dupla *MD5* para a *password* do utilizador é boa, mas poderia ser melhorada, como por exemplo com o uso de *Hashing with salt* [68], [69] ou encriptação e esteganografia da *password* com uma imagem [70].

O uso de gráficos circulares e de barras também seriam algo a ter em conta.

For fim, se a aplicação for um sucesso vai começar a ter muito tráfego para um único servidor, então é necessário reavaliar a replicação e a distribuição dos seus dados segundo o subcapítulo 2.4 de forma a contornar o problema.

[1] "2 Billion Consumers Worldwide to Get Smart(phones) by 2016 - eMarketer." [Online]. Available: http://www.emarketer.com/Article/2-Billion-Consumers-Worldwide-Smartphones-by-2016/1011694. [Accessed: 18-Feb-2015].

[2] "Mobile apps to hit >$70B revenue driven by explosion of diversity," *Digi-Capital Blog*, 2014. [Online]. Available: http://www.digi-capital.com/news/2014/04/mobile-apps-to-hit-70b-revenue-driven-by-explosion-of-diversity/#.VB9xuvldWQg. [Accessed: 22-Sep-2014].

[3] "API Definition from PC Magazine Encyclopedia." [Online]. Available: http://www.pcmag.com/encyclopedia/term/37856/api. [Accessed: 09-Mar-2015].

[4] "Android grabs record 85 percent smartphone share | PCWorld." [Online]. Available: http://www.pcworld.com/article/2460020/android-grabs-record-85-percent-smartphone-share.html. [Accessed: 09-Mar-2015].

[5] "Who's Winning, iOS or Android? All the Numbers, All in One Place | TIME.com." [Online]. Available: http://techland.time.com/2013/04/16/ios-vs-android/. [Accessed: 09-Mar-2015].

[6] "Android hits 83.6 percent marketshare while iOS, Windows and BlackBerry slide- The Inquirer." [Online]. Available: http://www.theinquirer.net/inquirer/news/2379036/android-hits-836-percent-marketshare-while-ios-windows-and-blackberry-slide. [Accessed: 09-Mar-2015].

[7] "Type Of Mobile Apps - Native App, Hybrid App, Web Applications." [Online]. Available: http://www.socialhunt.net/blog/types-of-mobile-app/. [Accessed: 09-Mar-2015].

[8] "Mobile Apps: Deciding Which App Type Is Best." [Online]. Available: http://techblog.constantcontact.com/software-development/best-app-type-for-mobile-apps/. [Accessed: 09-Mar-2015].

[9] "3 Basic Types of Mobile Event Apps." [Online]. Available: http://blog.omnipress.com/2012/11/basic-types-mobile-event-apps/. [Accessed: 09-Mar-2015].

[10] B. Plale and Y. Liu, "Survey of Publish Subscribe Event Systems," 2003.

[11] E. Anceaume, A. Datta, M. Gradinariu, and G. Simon, "Publish/subscribe scheme for mobile networks," pp. 74–81, 2002.

[12] G. Banavar, T. Chandra, B. Mukherjee, J. Nagarajarao, R. E. Strom, D. C. Sturman, I. B. M. T. J. Watson, S. Mill, and R. Road, "An Efficient Multicast Protocol for Content-Based Publish-Subscribe Systems," *Int. Conf. Distrib. Comput. Syst.*, 1999.

[13] Y. Jia, E. Bodanese, and J. Bigham, "Model checking of the reliability of publish/subscribe structure based system," in *2012 1st IEEE International Conference on Communications in China (ICCC)*, 2012, pp. 155–160.

[14] J. Nogueira, "An efficient multicast protocol for content-based publish-subscribe systems," 2006.

[15] A. S. TANENBAUM and D. J. WETHERALL, *Computer Networks*, FIFTH EDIT. 2011, p. 962.

[16] S. Tarkoma, *Publish/Subscribe Systems: Design and Principles*, First. 2012, p. 341.

[17] P. T. Eugster, P. a. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," *ACM Comput. Surv.*, vol. 35, no. 2, pp. 114–131, Jun. 2003.

[18] "lis3353 - Peer to Peer." [Online]. Available: http://lis3353.wikispaces.com/Peer+to+Peer. [Accessed: 14-Dec-2014].

[19] Y. Huang and H. Garcia-molina, "Publish / Subscribe in a Mobile Environment," no. section 2, pp. 643–652, 2004.

[20] C. Liu, Y. Liu, X. Ma, and J. Gao, "An Application Scheme of Publish/Subscribe System over Clustering Mobile Ad Hoc Networks," *2010 Int. Conf. Comput. Intell. Softw. Eng.*, pp. 1–4, Sep. 2010.

[21] M. Pandey and B. D. Chaudhary, "A Reconfigurable Distributed Broker Infrastructure for Publish Subscribe Based MANET," *2008 IEEE Int. Conf. Sens. Networks, Ubiquitous, Trust. Comput. (sutc 2008)*, pp. 361–366, Jun. 2008.

[22] U. Farooq, E. W. Parsons, and S. Majumdar, "Performance of Publish/Subscribe Middleware in Mobile Wireless Networks," 2004.

[23] C.-L. Hu and C.-A. Cho, "User-provided multimedia content distribution architecture in mobile and ubiquitous communication networks," *J. Netw. Comput. Appl.*, vol. 34, no. 1, pp. 121–136, Jan. 2011.

[24] T.-S. Chen, G.-J. Yu, and H.-J. Chen, "A framework of mobile context management for supporting context-aware environments in mobile ad hoc networks," *Proc. 2007 Int. Conf. Wirel. Commun. Mob. Comput. - IWCMC '07*, p. 647, 2007.

[25] C. R. Ozansoy, A. Zayegh, and A. Kalam, "The Real-Time Publisher/Subscriber Communication Model for Distributed Substation Systems," *IEEE Trans. Power Deliv.*, vol. 22, no. 3, pp. 1411–1423, Jul. 2007.

[26] P. Pinto and L. Bernardo, "Redes Integradas de Telecomunicações I," 2013.

[27] H. L. Tan, C. C. Wong, J. K. N. M, S. Hoh, P. Sentral, and J. S. Sentral, "Moving Towards an Era of Creativity and Growth : Co-creating with Customers," *IEEE*, p. 8, 2008.

[28] L. Duan, T. Kubo, K. Sugiyama, J. Huang, T. Hasegawa, and J. Walrand, "Incentive Mechanisms for Smartphone Collaboration in Data Acquisition and Distributed Computing," no. 412710, pp. 1701–1709, 2012.

[29] K. Li and T. C. Du, "Building a targeted mobile advertising system for location-based services," *Decis. Support Syst.*, vol. 54, no. 1, pp. 1–8, Dec. 2012.

[30] K. Palanivel, V. Amouda, and S. Kuppuswami, "Publisher-subscriber: An agent system for notification of versions in OODBs," *2009 Int. Conf. Intell. Agent Multi-Agent Syst.*, pp. 1–6, Jul. 2009.

[31] J.-Y. Huang and C.-Y. Gau, "Modelling and designing a low-cost high-fidelity mobile crane simulator," *Int. J. Hum. Comput. Stud.*, vol. 58, no. 2, pp. 151–176, Feb. 2003.

[32] "Geoinformatics Laboratory, School of Information Science." [Online]. Available: http://gis.sis.pitt.edu/diagrams.php?diagram=lbs. [Accessed: 10-Jan-2015].

[33] Y. Liu, E. Wilde, and U. C. Berkeley, "Personalized Location-Based Services," *iConference*, vol. February, pp. 496–502, 2011.

[34] B. Rao and L. Minakakis, "Evolution of Mobile Location-based Services," vol. 46, no. 12, pp. 61–65, 2003.

[35] W. J. Buchanan, Z. Kwecka, and E. Ekonomou, "A Privacy Preserving Method Using Privacy Enhancing Techniques for Location Based Services," *Mob. Networks Appl.*, vol. 18, no. 5, pp. 728–737, Apr. 2012.

[36] C.-S. Yang, P.-W. Tsai, M.-Y. Liao, C.-C. Huang, and C. E. Yeh, "Location-Based Mobile Multimedia Push System," *2010 Int. Conf. Cyber-Enabled Distrib. Comput. Knowl. Discov.*, pp. 181–184, Oct. 2010.

[37] L. Vogel, "Android Development - Tutorial," pp. 1–63, 2014.

[38] G. B. Creus and M. Kuulusa, "Mobile Phone Programming Android Advanced UI Features," pp. 449–462, 2007.

[39] G. B. Creus and M. Kuulusa, "Mobile Phone Programming Android PLatform Basics," pp. 449–462, 2007.

[40] G. B. Creus and M. Kuulusa, "Mobile Phone Programming Android UI," pp. 449–462, 2007.

[41] M. Phoneprogramming, "Persistent data storage Data storage intro."

[42] A. N. Communication, "Quick quizzes during the course."

[43] G. B. Creus and M. Kuulusa, "Mobile Phone Programming," pp. 449–462, 2007.

[44] M. P. Programming, "Communication of application components Where are we now ?," pp. 1–64.

[45] M. P. Programming, "Quick quizzes during the course," pp. 1–90.

[46] M. P. Programming, "Services and ContentProvider," pp. 1–34.

[47] M. P. Programming, “Widgets , Native Development and Extras.”

[48] “Dashboards | Android Developers.” [Online]. Available: https://developer.android.com/about/dashboards/index.html. [Accessed: 09-Mar-2015].

[49] “LocationManager | Android Developers.” [Online]. Available: http://developer.android.com/reference/android/location/LocationManager.html. [Accessed: 09-Mar-2015].

[50] “LocationListener | Android Developers.” [Online]. Available: http://developer.android.com/reference/android/location/LocationListener.html. [Accessed: 09-Mar-2015].

[51] “Google Play Services | Android Developers.” [Online]. Available: https://developer.android.com/google/play-services/index.html. [Accessed: 10-Mar-2015].

[52] “API Keys - Google APIs Client Library for Python — Google Developers.” [Online]. Available: https://developers.google.com/api-client-library/python/guide/aaa_apikeys. [Accessed: 10-Mar-2015].

[53] “Saving Data in SQL Databases | Android Developers.” [Online]. Available: http://developer.android.com/training/basics/data-storage/databases.html. [Accessed: 10-Mar-2015].

[54] “Android SQLite database and content provider - Tutorial.” [Online]. Available: http://www.vogella.com/tutorials/AndroidSQLite/article.html. [Accessed: 10-Mar-2015].

[55] “Android SQLite Database Tutorial.” [Online]. Available: http://www.androidhive.info/2011/11/android-sqlite-database-tutorial/. [Accessed: 10-Mar-2015].

[56] “JSON.” [Online]. Available: http://json.org/. [Accessed: 10-Mar-2015].

[57] “JSON Schema: core definitions and terminology.” [Online]. Available: http://json-schema.org/latest/json-schema-core.html#anchor4. [Accessed: 10-Mar-2015].

[58] “JSON (JavaScript Object Notation) Definition.” [Online]. Available: http://techterms.com/definition/json. [Accessed: 10-Mar-2015].

[59] “HTTP Requests.” [Online]. Available: http://www.tutorialspoint.com/http/http_requests.htm. [Accessed: 10-Mar-2015].

[60] “The HTTP Protocol As Implemented In W3.” [Online]. Available: http://www.w3.org/Protocols/HTTP/AsImplemented.html. [Accessed: 10-Mar-2015].

[61] “HTTP/1.1: Method Definitions.” [Online]. Available: http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html. [Accessed: 10-Mar-2015].

[62] “HyperText Transfer Protocol.” [Online]. Available: http://www.w3.org/History/19921103-hypertext/hypertext/WWW/Protocols/HTTP.html. [Accessed: 10-Mar-2015].

[63] “Hypertext Transfer Protocol -- HTTP/1.1.” [Online]. Available: http://www.w3.org/Protocols/rfc2616/rfc2616.html. [Accessed: 10-Mar-2015].

[64] “ADT Plugin Release Notes | Android Developers.” [Online]. Available: http://developer.android.com/tools/sdk/eclipse-adt.html. [Accessed: 24-Feb-2015].

[65] “WampServer, the web development platform on Windows - Apache, MySQL, PHP.” [Online]. Available: http://www.wampserver.com/en/. [Accessed: 24-Feb-2015].

[66] “Welcome to NetBeans.” [Online]. Available: https://netbeans.org/. [Accessed: 25-Feb-2015].

[67] “achartengine-1.1.0.jar - achartengine - The AChartEngine 1.1.0 binary build library. - Charting library for Android - Google Project Hosting.” [Online]. Available: https://code.google.com/p/achartengine/downloads/detail?name=achartengine-1.1.0.jar&can=2&q=. [Accessed: 17-Mar-2015].

[68] “Secure Salted Password Hashing - How to do it Properly.” [Online]. Available: https://crackstation.net/hashing-security.htm. [Accessed: 18-Mar-2015].

[69] S. Srinivasa and R. Mothukuri, “Pecking Order Hash – A New Hashing Algorithm,” vol. 1, no. 9, pp. 171–175, 2015.

[70] M. Hussain, A. Wahid, A. Wahab, and I. Batool, “Secure Password Transmission for Web Applications over Internet using Cryptography and Image Steganography,” vol. 9, no. 2, pp. 179–188, 2015.

[71] “global-search-icon.jpg (1280×1024).” [Online]. Available: http://www.psdgraphics.com/file/global-search-icon.jpg. [Accessed: 05-Mar-2015].

[72] “graph.png (800×800).” [Online]. Available: http://www.clipartlord.com/wp-content/uploads/2012/11/graph.png. [Accessed: 05-Mar-2015].

[73] “135126634414019667 45login-button 4 ti ppu-hi.png (600×175).” [Online]. Available: http://www.clker.com/cliparts/7/f/d/4/135126634414019667 45login-button 4 ti ppu-hi.png. [Accessed: 05-Mar-2015].

[74] “Wallpapers-room_com___Digital_Aurora_Dark-Blue_by_Martin-Matjulski_1920x1200.jpg (1920×1200).” [Online]. Available: http://content.wallpapers-room.com/resolutions/1920x1200/D/Wallpapers-room_com___Digital_Aurora_Dark-Blue_by_Martin-Matjulski_1920x1200.jpg. [Accessed: 05-Mar-2015].

[75] “IrRADIARE, Science for evolution.” [Online]. Available: http://ww2.irradiare.com/. [Accessed: 05-Mar-2015].

[76] “Faculdade de Ciências e Tecnologia / Universidade Nova de Lisboa.” [Online]. Available: http://www.fct.unl.pt/. [Accessed: 05-Mar-2015].

[77] “socu icon.gif (423×444).” [Online]. Available: http://auctivation.com/picks/gnu/socu icon.gif. [Accessed: 05-Mar-2015].

Tabela 32 Tabela resumo dos diferentes sistemas *publisher/subscriber* [16].

System	Subscription model	Infrastructure	Routing
Gryphon, Rebeca, SIENA, JEDI, Elvin, XSIENA	Content	Brokers, overlay	Filtering, filter aggregation, various extensions
Padres	Content	Brokers, overlay	Filtering, dynamic configuration, load balancing
REDS	Subscription, request-reply, pluggable	Pluggable modules: structured or unstructured	Pluggable modules: wide-area, mobile ad hoc network, reconfiguration of topology
GREEN	Various (pluggable components)	Pluggable modules: structured or unstructured	Pluggable modules: wide-area, mobile ad hoc network
Fuego	Content	Various, federated clusters	Rendezvous
STEAM	Subject, proximity	Proximity-based Group Communication Service	Proximity based
ECho and JECho	Channel (ECho) and stateful modulators (JECho)	Brokers, overlay	Filtering, load balancing and mobility (JECho)

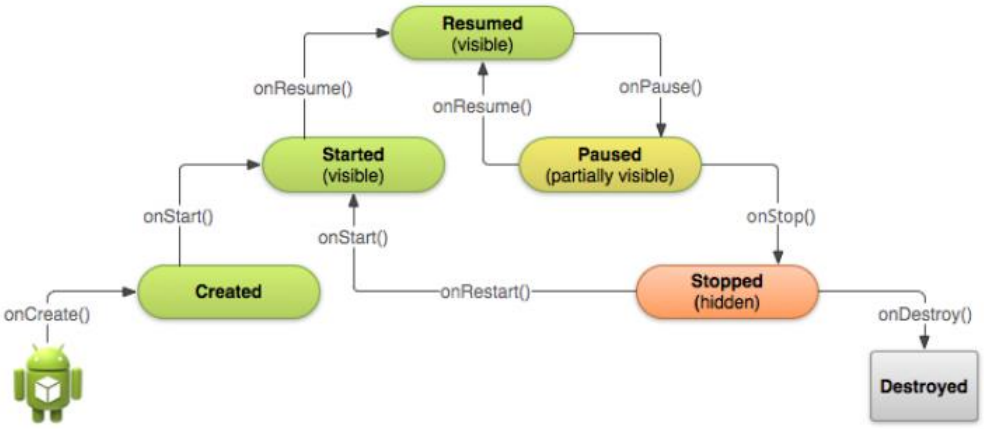


Figura 5-1 Ciclo de vida de uma actividade[37]–[47].

Tabela 33 – Lista com as figuras utilizadas pela aplicação móvel.

Figura5-2 *displaymapbutton.png*

Figura 5-3 *edit_button.png*

Figura5-4 g*lobal_search_icon.png*[71]

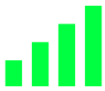


Figura5-5 *graph_image_final.png*

Figura5-6 *graph_image_grey.png*

Figura 5-7 *graph_image.png*[72]

Figura 5-8 *ic_lancher.png*

Figura5-9 *image_image_grey.png*

Figura 5-10 *image_image.png*

Figura 5-11 *login_green.png*[73]

Figura 5-12 *quick_search.png*

Figura 5-13 *refresh_button.png*

Figura 5-14 *register_button.png*

Figura 5-15 *return_button.png*

Figura5-16 *rsz_dark_blue_and_green_wallpaper_2_converted.jpg*[74]

Figura5-17 *rsz_first_finder_26_jan_azul.jpg*

Figura 5-18 *rsz_irradiare_logo.png*[75]

Figura5-19 *rsz_logo_nova.png*[76]

Figura 5-20 *text_image_final.png*

Figura5-21 t*ext_image_gray.png*

Figura 5-22 *text_image.png*[77]